Data Race Report

Yulei Sui and Peng Di

May 13th 2014

1 Thread API

1.1 Thread Management

The following APIs create/destroy/join/detach a thread

1.1.1 Create and Terminating a Thread:

- `pthread_create(thread, attr, start_routine, arg)`

  A thread (including main) spawns a new thread to call a *start_routine* with an argument *arg*
- `pthread_exit (status)` A thread returns normally from its starting rountine, or call pthread_exit to early terminte itself.
- `exec() or exit()` The entire process is terminated

1.1.2 Hard to be statically modeled:

The following APIs are hard to model due to its

- `pthread_cancel(thread)`

  One thread is terminated by other thread.
- `pthread_join(threadid,status)`

  Blocks the calling thread until the specified *threadid* thread (must be created joinable) terminates. Clean up any resources associated with the thread.
- `pthread_detach(threadid)` The thread are never going to join with the spawner thread again This allows the pthread library to know whether it can immediately dispose of the thread resources once the thread exits (the detached case) or whether it must keep them around because it may later call pthread_join on the spawner thread.

  Multiple pthread_detach calls on the same target thread causes an error

1.2 Mutexes

1.2.1 Creating and Destroying Mutexes

A mutex variable and its attributes need to be initialized and destroyed when they are not used.

- `pthread_mutex_init (mutex,attr)`

  A mutex variable must be initialized before they can be used

- `pthread_mutex_destroy (mutex)`

  If a mutex variable is destroyed, then it can not be used later

- `pthread_mutexattr_init (attr)`

  A mutex variable attribute must be initialized before they can be used

- `pthread_mutexattr_destroyed (attr)`

  If a mutex attribute is destroyed, then it can not be used later

1.2.2 Locking and Unlocking Mutexes

Locking and unlocking a mutex variable is used to protect the code segments to avoid data race.

- `pthread_mutex_lock (mutex)`

  Acquire a lock on a mutex variable by a thread, if the mutex is already locked by another thread, the call will be blocked until the mutex variable is released.

- `pthread_mutex_trylock (mutex)` Attempt to lock a mutex variable, if the mutex is already locked, return a "busy" to prevent deadlock

- `pthread_mutex_unlock (mutex)` Release a mutex by the owning thread. An error occur if (1) mutex was already unlocked (2) the mutex is owned by another thread.

1.3 Condition Variables

1.3.1 Creating and Destroying Condition Variables

Similar as mutex variables, conditional variables must be initialized first and destroyed when they are not used any more.

- `pthread_cond_init (condition, attr)`

  Create a condition variable

- `pthread_cond_destroy (condition, attr)`

  Destroy a condition variable

- `pthread_cond_init (attr)`

  Initialize a condition variable attribute

- `pthread_cond_destroy (attr)`

  Destroy a condition variable attribute

1.3.2 Waiting and Signaling

Condition variable provide another way for threads to synchronize. It synchronize based on actual values of data instead of controling thread access to data

- `pthread_cond_wait (condition,mutex)`

  Block the calling thread until the *condition* is signalled. It automatically and atomically releases mutex and wait for the signal. After signal is received the mutex will be automatically and atomically locked

- `pthread_cond_signal (condition)`

  Signal (weak up) another thread which is waiting on the condition variable. It should be called after *mutex* is locked and unlock *mutex* in order for `pthread_cond_wait` to complete

- `pthread_cond_broadcast (condition)`

  Broadcast signal to weak up more than one thread

2 Statistic From Existing Tools

3 Dynamic Data Race Detection

This section introduces dynamic data race algorithm implemented in ThreadSanitizer (`TSan`).

$$\texttt{[FORK]}\quad \frac{fork(i \triangleright j)}{C_i := inc_i(C_i) \quad C_j := C_i \sqcup C_j}$$

$$\texttt{[JOIN]}\quad \frac{join(i \triangleleft j)}{C_i := C_i \sqcup C_j \quad C_j := inc_j(C_j)}$$

$$\texttt{[ACQUIRE]}\quad \frac{acquire(i, l)}{C_i := C_i \sqcup C_l}$$

$$\texttt{[RELEASE]}\quad \frac{acquire(i, l)}{C_i := inc_i(C_i) \quad C_l := C_i}$$

$$\texttt{[LOAD]}\quad \frac{i : p = *q \quad E_w(\sigma(*q)) \sqsubseteq C_i}{\sigma(*q) := [i, load, addr(*q), C_i]}$$

$$\texttt{[STORE]}\quad \frac{i : *p = q \quad E_r(\sigma(*p)) \sqsubseteq C_i \quad E_w(\sigma(*p)) \sqsubseteq C_i}{\sigma(*p) := [i, store, addr(*p), C_i]}$$

	Create	Join	Lock	Trylock	Unlock	Cancel	Exit	Detach
parsec.blackscholes.res	1	1	0	0	0	0	0	0
parsec.bodytrack.res	1	1	1	1	1	0	0	0
parsec.ferret.res	1	1	3	0	4	1	0	0
parsec.fluidanimate.res	1	1	6	2	10	0	0	0
parsec.raytrace.res	2	0	24	0	30	0	0	0
parsec.swaptions.res	1	1	0	0	0	0	0	0
parsec.vips.res	14	14	56	14	70	0	14	0
parsec.x264.res	2	4	4	0	4	0	0	0
parsec.canneal.res	1	1	1	0	1	0	0	0
parsec.dedup.res	5	5	6	0	7	0	0	0
parsec.streamcluster.res	1	1	4	2	8	0	0	0
parsec.glib.res	43	43	0	43	0	0	43	0
parsec.libxml2.res	2	2	132	0	165	0	0	0
parsec.zlib.res	0	0	0	0	0	0	0	0
parsec.tools.yasm.res	0	0	0	0	0	0	0	0
splash.res	16	16	133	0	148	0	0	0
splash2.barnes.res	1	1	12	0	12	0	0	0
splash2.fmm.res	1	1	41	0	41	0	0	0
splash2.ocean_cp.res	1	1	24	0	24	0	0	0
splash2.ocean_ncp.res	1	1	23	0	23	0	0	0
splash2.radiosity.res	3	3	38	0	50	0	0	0
splash2.raytrace.res	1	1	13	0	16	0	0	0
splash2.volrend.res	1	1	23	0	23	0	0	0
splash2.water_nsquared.res	1	1	18	0	18	0	0	0
splash2.water_spatial.res	1	1	19	0	19	0	0	0
splash2.cholesky.res	1	1	11	0	11	0	0	0
splash2.fft.res	1	1	8	0	8	0	0	0
splash2.lu_cb.res	1	1	6	0	6	0	0	0
splash2.lu_ncb.res	1	1	6	0	6	0	0	0
splash2.radix.res	1	1	13	0	13	0	0	0
splash2x.barnes.res	1	1	14	0	14	0	0	0
splash2x.fmm.res	1	1	46	0	46	0	0	0
splash2x.ocean_cp.res	1	1	24	0	24	0	0	0
splash2x.ocean_ncp.res	1	1	23	0	23	0	0	0
splash2x.radiosity.res	3	3	74	0	103	0	0	0
splash2x.raytrace.res	1	1	15	0	20	0	0	0
splash2x.volrend.res	1	1	22	0	23	0	0	0
splash2x.water_nsquared.res	1	1	20	0	20	0	0	0
splash2x.water_spatial.res	1	1	20	0	20	0	0	0
splash2x.cholesky.res	1	1	12	0	13	0	0	0
splash2x.fft.res	1	1	8	0	8	0	0	0
splash2x.lu_cb.res	1	1	7	0	7	0	0	0
splash2x.lu_ncb.res	1	1	7	0	7	0	0	0
splash2x.radix.res	1	1	13	0	13	0	0	0

Table 1: Pthread API methods invocations on LLVM IR of Parsec/Splash2

Program	Reads Count	Writes Count	Ignored Reads Count
parsec.blackscholes	236	120	25
parsec.bodytrack	16993	14265	803
parsec.canneal	1669	1364	94
parsec.dedup	2301	1440	1576
parsec.ferret	10763	4110	791
parsec.fluidanimate	1567	601	132
parsec.raytrace	669	561	19
parsec.streamcluster	1142	456	82
parsec.swaptions	1342	607	162
parsec.vips	87803	36611	8976
parsec.x264	48405	16675	2849
splash2.barnes	2188	1222	395
splash2.cholesky	7086	2530	930
splash2.fft	1048	387	88
splash2.fmm	3868	1447	237
splash2.lu_cb	1097	408	81
splash2.lu_ncb	840	303	73
splash2.ocean_cp	9531	2301	410
splash2.ocean_ncp	5465	1381	301
splash2.radiosity	5250	1917	269
splash2.radix	777	354	120
splash2.raytrace	6049	2368	359
splash2.volrend	8739	3723	1009
splash2.water_nsquared	2188	703	181
splash2.water_spatial	2437	833	211

Table 2: TSan Instrumentation Statistics (under compiler option -flto -O0)

Program	Warnings Count	Time (Sec)
parsec.blackscholes	0	63.77
parsec.bodytrack	59	366.35
parsec.canneal	41934	464.28
parsec.dedup	0	1985.48
parsec.fluidanimate	1	593.70
parsec.streamcluster	1294	5341.89
parsec.swaptions	0	20.47

Table 3: Valgrind data race report for parsec benchmarks using simulation inputs with 16 threads (under compiler option -flto -O0)

Program	Race warnings	Total Time
parsec.blackscholes	0	1m2.924s
parsec.bodytrack	63	4m55.364s
parsec.canneal	42041	7m36.824s
parsec.dedup	0	27m16.520s
parsec.ferret	0	0
parsec.fluidanimate	0	10m8.280s
parsec.streamcluster	1316	84m12.944s
parsec.swaptions	0	4m6.676s
parsec.vips	10384	19m54.096s
parsec.x264	920146	32m39.956s
splash2.barnes	0	0m1.480s
splash2.cholesky	0	0m0.784s
splash2.fft	0	0m0.768s
splash2.fmm	0	0m0.840s
splash2.lu_cb	0	0m0.856s
splash2.lu_ncb	0	0m0.876s
splash2.ocean_cp	0	0m0.960s
splash2.ocean_ncp	0	0m0.904s
splash2.radiosity	0	0m3.780s
splash2.radix	0	0m0.828s
splash2.raytrace	0	0m0.864s
splash2.volrend	0	0m0.776s
splash2.water_nsquared	0	0m0.952s
splash2.water_spatial	0	0m0.944s

Table 4: Valgrind data race report for parsec benchmarks using simulation inputs with 16 threads (under compiler option -flto -O0)

	Reads	Writes	Same thread	Another thread	Thread created	Lock	Unlock
parsec.blackscholes	718341129	555240430	1272271821	69348414	5	0	0
parsec.bodytrack	18803218920	10620341322	45096163095	1108277571	6	27774	27774
parsec.dedup	3012014550	3449596370	6703978198	239550770	15	548504	548504
*parsec.ferret							
parsec.fluidanimate	15499239043	9004451244	34626646787	871973803	5	4402100	4402100
parsec.raytrace	37407156989	27428638310	75372855802	414	5	22	22
parsec.streamcluster	55483706359	8344088145	62201163149	7896765779	9	115704	115704
parsec.swaptions	21020223624	4565945891	36573840102	940035163	5	0	0
parsec.vips	22140584399	8850071396	46078131202	2399680437	7	25655	25655
parsec.x264	136435977201	42430536547	234983597568	631911254	256	17200	17200
splash2.barnes	1492825769	1009970816	2822122316	177324320	4	68979	68979
splash2.fft	37433910	13593099	72139263	824858	4	53	53
splash2.fmm	185527433	6644797	208207974	2312613	4	1112	1112
splash2.lu_cb	426484609	114526095	595486936	21304096	4	473	473
splash2.lu_ncb	416255595	111823474	595648821	19638964	4	473	473
splash2.ocean_cp	615112705	104846529	937666585	19155012	4	7132	7132
splash2.ocean_ncp	840380414	103749891	1113302578	158979926	4	6922	6922
splash2.radiosity	8181232624	3709922702	16578904682	609993639	4	1312971	1312971
splash2.radix	30107430	19796004	63532201	2285498	4	113	113
splash2.raytrace	205607190	57687515	271999733	18773175	4	76740	76740
splash2.volrend	29783924	11014570	39628364	7711532	4	2440	2440
splash2.water_nsquared	504756939	107400563	667540204	72817240	4	6353	6353
splash2.water_spatial	425754990	94789606	634509203	78060332	4	218	218

Table 5: TSan stats with sim inputs of 4 threads (-flto -O0)

	Reads	Writes	Same thread	Another thread	Thread created	Lock	Unlock
parsec.blackscholes	718346583	555243070	1232918874	167399503	17	0	0
parsec.bodytrack	18803361154	10620441669	44782600945	1277852605	18	30899	30899
parsec.dedup	3018304143	3451449822	6787565705	241635211	51	544447	544447
*parsec.ferret							
parsec.fluidanimate	15669557555	9037159698	35197084050	1628354229	17	14019836	14019836
parsec.raytrace	37407159075	27428639477	75372861710	957	17	58	58
parsec.streamcluster	66308089986	12096654115	80994784283	19631825712	33	636218	636218
parsec.swaptions	21020224056	4565946035	36568179789	954109112	17	0	0
parsec.vips	22140170982	8850230085	45939795987	2735925812	19	33436	33436
parsec.x264	135883526982	42252542277	232953230265	704000171	256	18820	18820
splash2.barnes	1687036207	1009982147	2906344087	434664545	16	69561	69561
splash2.fft	37578522	13653835	72036880	1176114	16	233	233
splash2.fmm	3145819	1108617	5171891	103893	16	122	122
splash2.lu_cb	428918065	115429803	574877592	71354298	16	2093	2093
splash2.lu_ncb	418564179	112595794	573463588	72797425	16	2093	2093
splash2.ocean_cp	626006740	108123775	942764602	40415870	16	31228	31228
splash2.ocean_ncp	842713849	104698138	1052225655	313167816	16	30298	30298
splash2.radiosity	8269300550	3739673019	15928772052	1314727824	16	1404503	1404503
splash2.radix	32011402	20306228	65360322	3826822	16	517	517
splash2.raytrace	208180416	58212811	270190700	40510280	16	76788	76788
splash2.volrend	31756753	11113044	38902946	15724634	16	3438	3438
splash2.water_nsquared	508451427	108307475	658687685	139604442	16	19325	19325
splash2.water_spatial	425766066	94792780	629825838	89572231	16	926	926

Table 6: TSan stats with sim inputs of 16 threads (-flto -O0)

Program	Race warnings (4 threads)	Total Time (4 threads)	Race warnings (16 threads)	Total Time (16 threads)
parsec.blackscholes	0	4m20.216s	0	4m42.536s
parsec.bodytrack	9	125m17.432s	11	131m42.692s
*parsec.canneal	0	17m47.992s	0	18m22.012s
parsec.dedup	0	24m28.212s	0	26m54.492s
*parsec.ferret	0	0m10.064s	0	0m9.940s
parsec.fluidanimate	1	111m34.208s	1	123m2.620s
parsec.raytrace	2	382m34.544s	2	391m49.096s
parsec.streamcluster	17	224m54.612s	20	362m38.612s
parsec.swaptions	0	92m42.964s	0	97m28.616s
parsec.vips	1	127m7.076s	2	135m23.380s
parsec.x264	77	777m6.396s	67	808m52.088s
splash2.barnes	118	8m31.340s	100	10m58.704s
*splash2.cholesky	0	0m10.404s	0	0m9.640s
splash2.fft	0	0m10.832s	0	0m11.408s
splash2.fmm	145	1m14.956s	21	0m1.288s
splash2.lu_cb	0	1m37.004s	0	1m44.392s
splash2.lu_ncb	0	1m34.784s	0	1m42.596s
splash2.ocean_cp	1	2m33.472s	1	2m50.624s
splash2.ocean_ncp	2	3m25.632s	2	4m12.296s
splash2.radiosity	373	47m27.576s	0	50m51.992s
splash2.radix	0	0m10.756s	0	0m11.964s
splash2.raytrace	2	0m48.836s	3	0m53.248s
splash2.volrend	26	0m14.008s	37	0m25.092s
splash2.water_nsquared	0	1m51.428s	0	2m4.100s
splash2.water_spatial	0	1m40.940s	0	1m47.492s

Table 7: TSan race report using sim inputs with 4/16 threads (-flto -O0) on CORG-TESLA Note that parsec.canneal, splash2.cholesky can not run after TSan intrumentation.

Program	4 Threads		16 Threads	
	Native	TSan	Native	TSan
parsec.blackscholes	0m2.276s	0m18.464s	0m2.360s	0m22.624s
parsec.bodytrack	0m15.060s	8m35.412s	0m15.528s	9m28.368s
*parsec.canneal	0m10.080s	1m43.604s	0m10.008s	1m50.628s
parsec.dedup	0m25.124s	2m24.392s	0m32.548s	3m25.516s
*parsec.ferret	0m10.040s	0m9.952s	0m10.060s	0m10.016s
parsec.fluidanimate	0m12.200s	8m39.848s	0m13.240s	10m31.972s
parsec.raytrace	0m52.356s	27m59.244s	0m53.312s	28m36.068s
parsec.streamcluster	0m20.380s	19m14.276s	1m11.008s	119m6.380s
parsec.swaptions	0m14.528s	6m19.440s	0m15.260s	6m39.444s
parsec.vips	0m13.648s	8m13.408s	0m14.336s	9m13.280s
parsec.x264	1m30.504s	52m34.376s	1m33.928s	54m45.052s
splash2.barnes	0m1.604s	0m41.480s	0m2.664s	1m8.288s
*splash2.cholesky	0m0.044s	0m0.672s	0m2.028s	0m0.632s
splash2.fft	0m0.044s	0m0.720s	0m0.064s	0m0.768s
splash2.fmm	0m0.008s	0m15.560s	0m0.004s	0m0.360s
splash2.lu_cb	0m0.340s	0m6.788s	0m0.412s	0m7.428s
splash2.lu_ncb	0m0.356s	0m6.760s	0m0.432s	0m7.344s
splash2.ocean_cp	0m0.404s	0m10.104s	0m0.672s	0m14.968s
splash2.ocean_ncp	0m0.576s	0m13.584s	0m0.740s	0m42.112s
splash2.radiosity	0m5.508s	3m35.504s	0m8.364s	4m32.364s
splash2.radix	0m0.072s	0m0.788s	0m0.092s	0m0.996s
splash2.raytrace	0m0.168s	0m4.192s	0m0.204s	0m4.796s
splash2.volrend	0m0.036s	0m5.892s	0m1.776s	0m23.596s
splash2.water_nsquared	0m0.296s	0m8.024s	0m0.356s	0m10.432s
splash2.water_spatial	0m0.256s	0m7.080s	0m0.352s	0m8.648s

Table 8: Runtime comparison between native run and TSan under two configurations on CORG-TESLA

Program	Race warnings (4 threads)	Total Time (4 threads)	Race warnings (16 threads)	Total Time (16 threads)
parsec.blackscholes	0	0m16.970s	0	0m18.110s
parsec.bodytrack	9	7m26.560s	9	7m30.460s
*parsec.canneal	0	1m33.010s	0	1m33.500s
parsec.dedup	0	1m59.380s	0	5m33.010s
*parsec.ferret	0	0m10.160s	0	0m10.370s
parsec.fluidanimate	1	7m38.060s	1	8m15.010s
parsec.raytrace	2	24m1.880s	2	24m5.150s
parsec.streamcluster	17	14m46.680s	21	54m7.370s
parsec.swaptions	0	5m13.560s	0	5m16.480s
parsec.vips	0	0m0.010s	0	0m0.010s
parsec.x264	72	44m49.580s	62	44m46.310s
splash2.barnes	122	0m33.630s	123	0m46.010s
*splash2.cholesky	0	0m0.570s	0	0m0.530s
splash2.fft	0	0m0.570s	0	0m0.590s
splash2.fmm	151	0m3.340s	21	0m0.220s
splash2.lu_cb	0	0m5.440s	0	0m5.480s
splash2.lu_ncb	0	0m5.420s	0	0m5.490s
splash2.ocean_cp	1	0m8.770s	1	0m10.010s
splash2.ocean_ncp	2	0m12.310s	2	0m13.300s
splash2.radiosity	0	3m10.270s	536	3m23.740s
splash2.radix	0	0m0.600s	0	0m0.670s
splash2.raytrace	1	0m3.520s	1	0m3.710s
splash2.volrend	21	0m2.640s	29	0m7.140s
splash2.water_nsquared	0	0m6.670s	0	0m7.170s
splash2.water_spatial	0	0m5.980s	0	0m6.100s

Table 9: TSan race report using sim inputs with 4/16 threads (-flto -O0) Note that parsec.canneal, splash2.cholesky can not run after TSan intrumentation on JINGLE

Program	4 Threads		16 Threads	
	Native	TSan	Native	TSan
parsec.blackscholes	0m2.970s	0m16.970s	0m2.910s	0m18.110s
parsec.bodytrack	0m13.860s	7m26.560s	0m13.850s	7m30.460s
parsec.canneal	0m10.780s	1m33.010s	0m10.760s	1m33.500s
parsec.dedup	0m20.300s	1m59.380s	0m21.370s	5m33.010s
parsec.ferret	0m10.100s	0m10.160s	0m10.180s	0m10.370s
parsec.fluidanimate	0m10.690s	7m38.060s	0m11.360s	8m15.010s
parsec.raytrace	0m53.320s	24m1.880s	0m53.310s	24m5.150s
parsec.streamcluster	0m18.390s	14m46.680s	0m56.820s	54m7.370s
parsec.swaptions	0m15.180s	5m13.560s	0m15.140s	5m16.480s
parsec.vips	0m0.000s	0m0.010s	0m0.000s	0m0.010s
parsec.x264	1m15.690s	44m49.580s	1m15.620s	44m46.310s
splash2.barnes	0m1.410s	0m33.630s	0m2.890s	0m46.010s
splash2.cholesky	0m0.240s	0m0.570s	0m1.520s	0m0.530s
splash2.fft	0m0.030s	0m0.570s	0m0.030s	0m0.590s
splash2.fmm	0m0.010s	0m3.340s	0m0.000s	0m0.220s
splash2.lu_cb	0m0.190s	0m5.440s	0m0.190s	0m5.480s
splash2.lu_ncb	0m0.240s	0m5.420s	0m0.230s	0m5.490s
splash2.ocean_cp	0m0.280s	0m8.770s	0m0.330s	0m10.010s
splash2.ocean_ncp	0m0.410s	0m12.310s	0m0.430s	0m13.300s
splash2.radiosity	0m4.720s	3m10.270s	0m8.230s	3m23.740s
splash2.radix	0m0.040s	0m0.600s	0m0.050s	0m0.670s
splash2.raytrace	0m0.150s	0m3.520s	0m0.140s	0m3.710s
splash2.volrend	0m0.000s	0m2.640s	0m1.540s	0m7.140s
splash2.water_nsquared	0m0.260s	0m6.670s	0m0.280s	0m7.170s
splash2.water_spatial	0m0.210s	0m5.980s	0m0.220s	0m6.100s

Table 10: Runtime comparison between native run and TSan under two configurations on JINGLE