Splash-4: A Modern Benchmark Suite with Lock-Free Constructs

Eduardo José Gómez-Hernández[1] Juan M. Cebrian[1]
Stefanos Kaxiras[2] Alberto Ros[1]

[1] Computer Engineering Department
University of Murcia, Spain

[2] Department of Information Technology
Uppsala University, Sweden

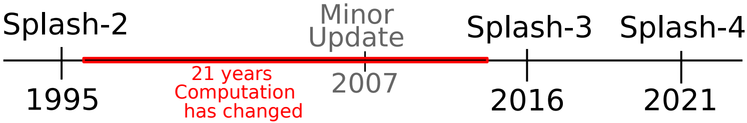
Overview

- The cornerstone for the performance evaluation is the benchmark suite
- Benchmark suites can misrepresent the performance characteristics
- Keeping up with architectural changes while maintaining the same workloads is a real challenge

OVERVIEW

- The cornerstone for the performance evaluation is the benchmark suite
- Benchmark suites can misrepresent the performance characteristics
- Keeping up with architectural changes while maintaining the same workloads is a real challenge
- We introduce Splash-4, a revised version of Splash-3 (an update on Splash-2), that introduces modern programming techniques to improve scalability on contemporary hardware

Motivation



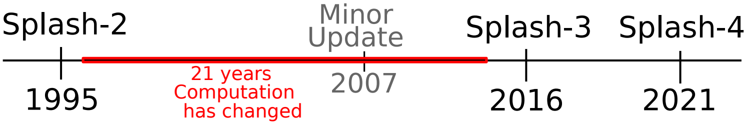
[1] Woo, Steven Cameron, et al, "The SPLASH-2 programs: Characterization and methodological considerations." ACM SIGARCH computer architecture news 23, 1995

[2] Venetis, Ioannis E., et al, "The Modified SPLASH-2", https://www.capsl.udel.edu//splash/ 2007

[3] Sakalis, Christos, et al, "Splash-3: A Properly Synchronized Benchmark Suite for Contemporary Research", ISPASS 2016

[4] Gómez-Hernández, Eduardo José et al, "Splash-4: Improving Scalability with Lock-Free Constructs", ISPASS 2021

Motivation



Splash-2

1995

21 years Computation has changed

Minor Update

2007

Splash-3

2016

Splash-4

2021

The first major parallel benchmark suite. Still in use (+5k cites)[1]

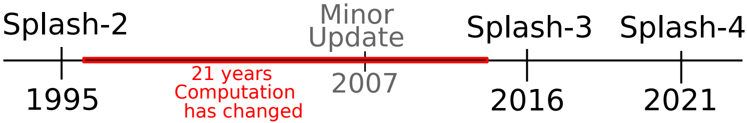
[1] Woo, Steven Cameron, et al, "The SPLASH-2 programs: Characterization and methodological considerations." ACM SIGARCH computer architecture news 23, 1995

[2] Venetis, Ioannis E., et al, "The Modified SPLASH-2", https://www.capsl.udel.edu//splash/ 2007

[3] Sakalis, Christos, et al, "Splash-3: A Properly Synchronized Benchmark Suite for Contemporary Research", ISPASS 2016

[4] Gómez-Hernández, Eduardo José et al, "Splash-4: Improving Scalability with Lock-Free Constructs", ISPASS 2021

Motivation



Splash-2

1995

Minor Update

2007

21 years Computation has changed

Splash-3

2016

Splash-4

2021

The first major parallel benchmark suite. Still in use (+5k cites)[1]

A small update that fixes bugs and updates the programming style[2]

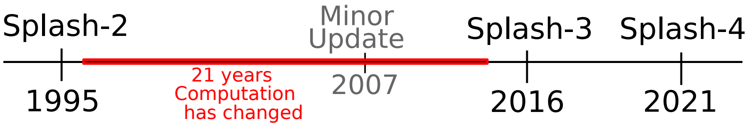
[1] Woo, Steven Cameron, et al, "The SPLASH-2 programs: Characterization and methodological considerations." ACM SIGARCH computer architecture news 23, 1995

[2] Venetis, Ioannis E., et al, "The Modified SPLASH-2", https://www.capsl.udel.edu//splash/ 2007

[3] Sakalis, Christos, et al, "Splash-3: A Properly Synchronized Benchmark Suite for Contemporary Research", ISPASS 2016

[4] Gómez-Hernández, Eduardo José et al, "Splash-4: Improving Scalability with Lock-Free Constructs", ISPASS 2021

MOTIVATION



The first major parallel benchmark suite. Still in use (+5k cites)[1]

A small update that fixes bugs and updates the programming style[2]

First major update that fixes data races and performance bugs[3]

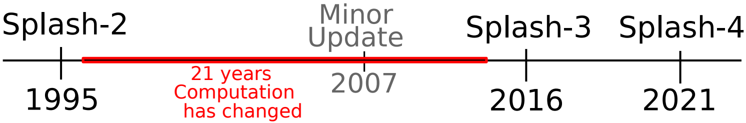
[1] Woo, Steven Cameron, et al, "The SPLASH-2 programs: Characterization and methodological considerations." ACM SIGARCH computer architecture news 23, 1995

[2] Venetis, Ioannis E., et al, "The Modified SPLASH-2", https://www.capsl.udel.edu//splash/ 2007

[3] Sakalis, Christos, et al, "Splash-3: A Properly Synchronized Benchmark Suite for Contemporary Research", ISPASS 2016

[4] Gómez-Hernández, Eduardo José et al, "Splash-4: Improving Scalability with Lock-Free Constructs", ISPASS 2021

Motivation



Splash-2

1995

Minor Update

2007

21 years Computation has changed

Splash-3

2016

Splash-4

2021

The first major parallel benchmark suite. Still in use (+5k cites)[1]

A small update that fixes bugs and updates the programming style[2]

First major update that fixes data races and performance bugs[3]

Current update, exploiting lockfree and atomic operations[4]

[1] Woo, Steven Cameron, et al, "The SPLASH-2 programs: Characterization and methodological considerations." ACM SIGARCH computer architecture news 23, 1995

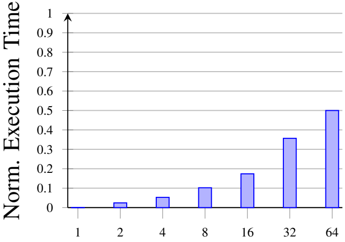
[2] Venetis, Ioannis E., et al, "The Modified SPLASH-2", https://www.capsl.udel.edu//splash/ 2007

[3] Sakalis, Christos, et al, "Splash-3: A Properly Synchronized Benchmark Suite for Contemporary Research", ISPASS 2016

[4] Gómez-Hernández, Eduardo José et al, "Splash-4: Improving Scalability with Lock-Free Constructs", ISPASS 2021

Motivation

- Splash-2 and Splash-3 are crafted using outdated programming techniques
- Previous works noticed that the default input sizes limit the scalability of some applications.
  - The computation between synchronization points is not substantially longer than the synchronization
  - Using larger datasets increases the execution time, and that is a problem when using simulation infrastructures



SPLASH-3

- Splash-3 exposes data races and performance bugs, and fixed them

- In the analysis done using GEMS (in a 64-core in-order multicore), benchmarks reach an speedup between 16× to 47×

- In our simulated environment (gem5-20 Intel's IceLake-like 64 out of order cores): the scalability stops between 16 and 32 cores, with an average speedup of 2.3×

- In our real hardware (64-Core AMD EPYC 7702P), scalability stops between 4 and 16 cores, with an average speedup of 4.7×

SPLASH-4

- Synchronization is done with combination of software locks, conditional variables, and barriers
- The main idea is to replace high-overhead synchronization operations with newer lightweight alternatives
- This translates into a performance improvement by expanding the architectural features that the benchmarks can exercise

Splash-4: LockFree and Atomics

- Modern ISAs typically provide a basic set of atomic operations that offer both atomicity and synchronization
- This basic set consists of atomic loads and stores, atomic read-modify-write (RMW) operations, and some atomic comparisons and exchange operations
- Typical hardware RMW atomics are only available for integer types

Volrend (adaptive.c.in:182/199)

Splash-3

```
/* Lock */
ALOCK(Global->QLock,local_node);
work = Global->Queue[local_node
    ][0];
Global->Queue[local_node][0] +=
    1;
AULOCK(Global->QLock,local_node);
```

Splash-4

```
/* Lock-free */
work = FETCH_ADD(Global->Queue[
    local_node][0], 1);
```

ALOCK/AULOCK → mutex lock/unlock

FETCH_ADD → atomic_fetch_and_add - Sequential Consistency

Splash-4: LockFree and Atomics

While&CAExch Construct

- Atomic Compare-and-Swap (CAS) and Atomic Compare-and-Exchange (CAExch) are type-agnostic
- They can be used to implement RMW operations for more complex underlying types

```
CAExch(ptr, oldValue, newValue);
```

- If $oldValue == (*ptr)$ then $(*ptr) = newValue$
- Else $oldValue = (*ptr)$

LOAD → Atomic Load

SPLASH-4: LOCKFREE AND ATOMICS

WHILE&CAEXCH CONSTRUCT

- Atomic Compare-and-Swap (CAS) and Atomic Compare-and-Exchange (CAExch) are type-agnostic
- They can be used to implement RMW operations for more complex underlying types

```
CAExch(ptr, oldValue, newValue);
```

- If $\mathit{oldValue} == (*\mathit{ptr})$ then $(*\mathit{ptr}) = \mathit{newValue}$
- Else $\mathit{oldValue} = (*\mathit{ptr})$

```
var oldValue = LOAD(ptr);
var newValue;
do {
  newValue = new;
} while (!CAExch(ptr, oldValue, newValue));
```

LOAD → Atomic Load

Ocean (multi.c.in:90)

Atomic Max

Splash-3

```
/* Lock */
LOCK(locks->error_lock)
if (local_err > multi->err_multi)
      {
  multi->err_multi = local_err;
}
UNLOCK(locks->error_lock)
```

Splash-4

```
/* Lock-free */
double expected = LOAD(multi->
    err_multi);
do {
  if (local_err <= expected)
    break;
} while (!CAExch(multi->err_multi
    , expected, local_err));
```

LOCK/UNLOCK → mutex lock/unlock
LOAD → Atomic Load - Sequential Consistency

Splash-4: Critical Section Splitting

- Splash-4 uses lock-free constructs that manage a single address and replace to critical sections that modify a single address
- Splitting a larger critical section that modifies more than one address would enable the use of lock-free constructs in more cases
- Unfortunately, splitting large critical sections, is not possible in the general case
- Many Splash-3 critical sections, (atomic) updates of independent variables are grouped together in larger critical sections for **no apparent reason**

WATER (POTENG.C.IN 159/253)

- Atomic Double Floating Point Addition (FETCH_AND_ADD_DOUBLE)

```
double oldValue = LOAD(ptr);
double newValue;
do {
 newValue = oldValue + addition;
} while (!CAExch(ptr, oldValue, newValue));
```

LOCK/UNLOCK → mutex lock/unlock

LOAD → Atomic Load - Sequential Consistency

Water (poteng.c.in 159/253)

- Atomic Double Floating Point Addition (FETCH_AND_ADD_DOUBLE)

```
double oldValue = LOAD(ptr);
double newValue;
do {
 newValue = oldValue + addition;
} while (!CAExch(ptr, oldValue, newValue));
```

Splash-3

```
/* Lock */
LOCK(gl->PotengSumLock);
*POTA = *POTA + LPOTA;
*POTR = *POTR + LPOTR;
*PTRF = *PTRF + LPTRF;
UNLOCK(gl->PotengSumLock);
```

LOCK/UNLOCK → mutex lock/unlock
LOAD → Atomic Load - Sequential Consistency

Splash-4

```
/* Lock-free */
FETCH_AND_ADD_DOUBLE(POTA, LPOTA)
     ;
FETCH_AND_ADD_DOUBLE(POTR, LPOTR)
     ;
FETCH_AND_ADD_DOUBLE(PTRF, LPTRF)
     ;
```

SPLASH-4: SENSE-REVERSING CENTRALIZED BARRIER

- The execution time between barriers is fairly short
- To minimize the overhead, we implement a sense-reversing barrier
- This construct is optimized for short waiting times, except when oversubscribing threads

LOAD → Atomic Load - Sequential Consistency
STORE → Atomic Store - Sequential Consistency

Splash-4: Sense-Reversing Centralized Barrier

- The execution time between barriers is fairly short
- To minimize the overhead, we implement a sense-reversing barrier
- This construct is optimized for short waiting times, except when oversubscribing threads

```
local_sense = !local_sense;
if (atomic_fetch_sub(&(count), 1) == 1) {
    count = threads;
  STORE(sense, local_sense);
} else {
  do {} while (LOAD(sense) != local_sense);
}
```

LOAD → Atomic Load - Sequential Consistency
STORE → Atomic Store - Sequential Consistency

SPLASH-4: SUMMARY

- A total of 50 critical sections (33%) has been transformed.
  - 27 critical sections (18%) were converted into C11 atomics
  - 18 critical sections (12%) were converted into CAS-construct (without splitting)
  - Splitting Critical sections allowed 5 more critical sections (3%) to be converted into 15 CAS-construct

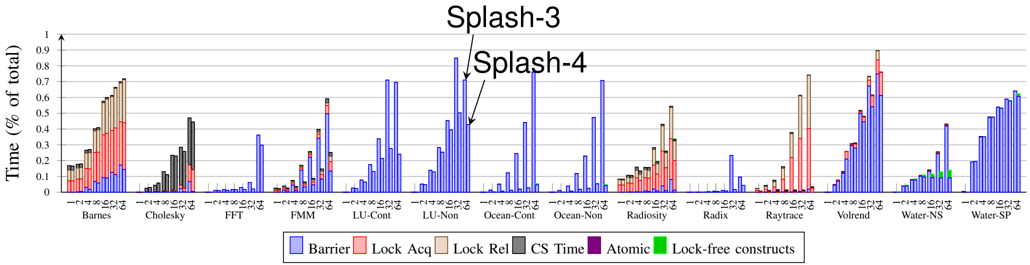
Evaluation: Methodology

- Measurements account for the region of interest (ROI)
- We used the recommended inputs from Splash-3 for all the executions

- The hardware used is an AMD Epyc 7702P CPU

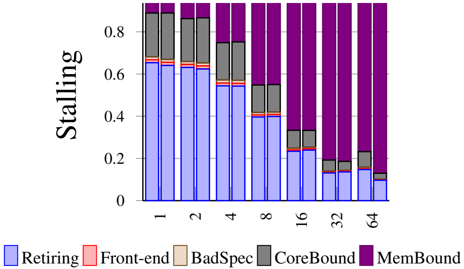
- The simulated machine (in gem5-20) is mimicking an Intel’s IceLake processor
  - Measurements reset stats after a warm-up period
    (as suggested by the original Splash-2 developers)

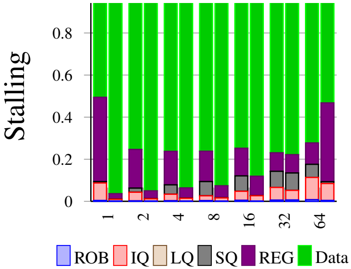
Evaluation: Simulator Overhead Breakdown



- Most of the cases the execution time is dominated by the barriers
- Raytrace gets a huge overhead reduction due to the lock-free constructs

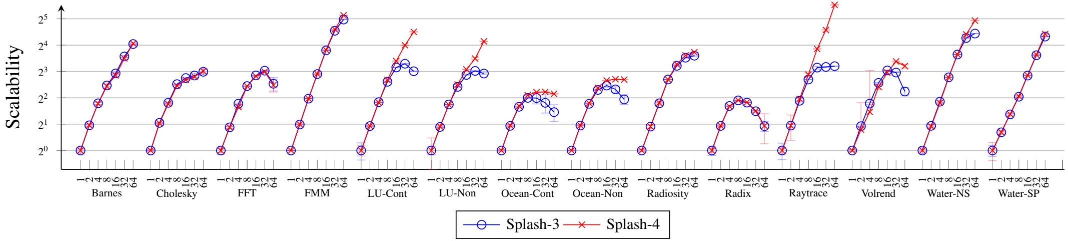
Evaluation: Top-Down (Ocean-Cont)





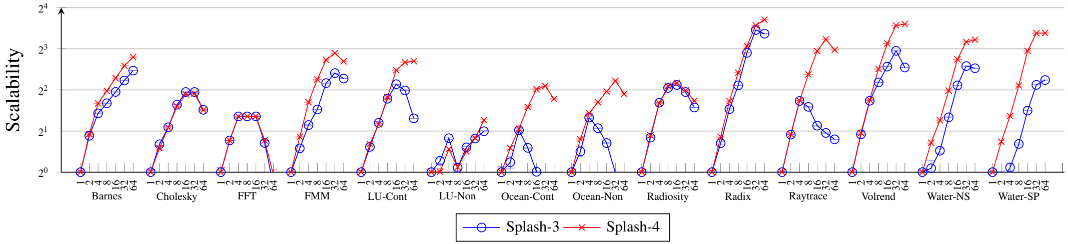
- Top-Down groups several hardware counters to understand easier the reason of the stalling
- In Splash-4 we observe an increase on the back-end pressure

Evaluation: Simulator Scalability



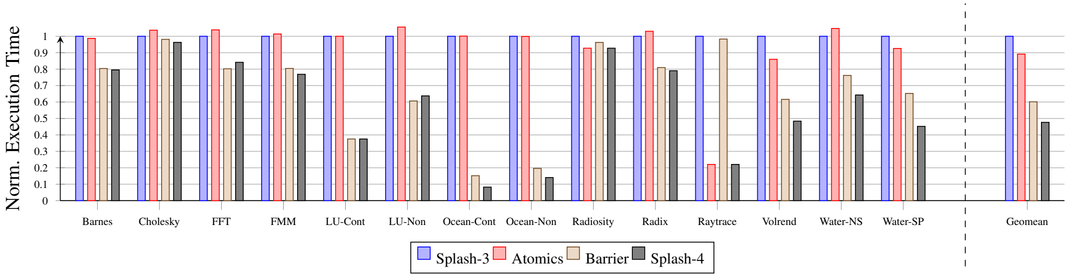
- In general, scalability improvement is moved from 16 to 32

Evaluation: Real Machine Scalability



- In general, scalability improvement is moved from 4 to 16

Evaluation: Real Machine 64 Threads Summary



- With the lock-free constructs the execution time is reduced 11%
- With the new barriers the execution time is reduced 40%
- The combined reduction is 52%

Conclusions

- We presented The Splash-4 Benchmark Suite
- We performed a detailed performance analysis comparing with Splash-3

Conclusions

- We presented The Splash-4 Benchmark Suite
- We performed a detailed performance analysis comparing with Splash-3
- Our study shows:
  - A significant improvement on scalability
  - Execution time is reduced significantly on the simulated and real hardware

CONCLUSIONS

- We presented The Splash-4 Benchmark Suite
- We performed a detailed performance analysis comparing with Splash-3
- Our study shows:
  - A significant improvement on scalability
  - Execution time is reduced significantly on the simulated and real hardware
- In summary:
  - The Splash benchmark suite is still a cornerstone in computer architecture research
  - It should be updated to be able to exploit modern hardware features

Conclusions

- We presented The Splash-4 Benchmark Suite
- We performed a detailed performance analysis comparing with Splash-3
- Our study shows:
  - A significant improvement on scalability
  - Execution time is reduced significantly on the simulated and real hardware
- In summary:
  - The Splash benchmark suite is still a cornerstone in computer architecture research
  - It should be updated to be able to exploit modern hardware features

Splash-4 removes some of the synchronization overheads, showing the hidden reasons that prevents applications from scaling, leaving a door open for researchers to further improve their designs

SPLASH-4: A MODERN BENCHMARK SUITE WITH LOCK-FREE CONSTRUCTS

Eduardo José Gómez-Hernández[1] Juan M. Cebrian[1]
Stefanos Kaxiras[2] Alberto Ros[1]

eduardojose.gomez@um.es

Thank you for your attention!

ECHO, ERC Consolidator Grant (No 819134)
Spanish Ministerio de Economía, Industria y Competitividad – Agencia Estatal de Investigación (ERC2018-092826)

INDEPENDENT VARIABLES

- During the possible window of execution time of the critical section, all variables cannot be related on any way, even not giving information from one to another

```
LOCK(lock);
ptr = NULL;
some_random_boolean = false;
UNLOCK(lock);
```

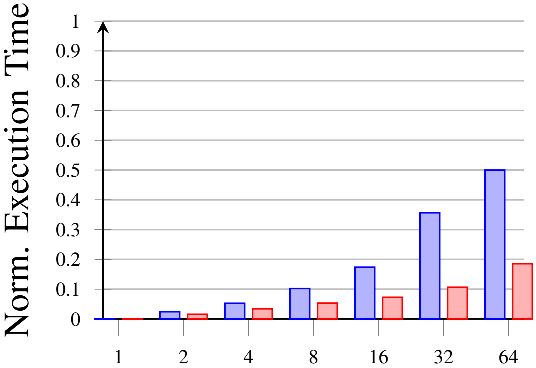
Independent variables

- During the possible window of execution time of the critical section, all variables cannot be related on any way, even not giving information from one to another

```
LOCK(lock);
ptr = NULL;
some_random_boolean = false;
UNLOCK(lock);
```

```
LOCK(lock);
if (some_random_boolean) {
    local = *ptr;
}
UNLOCK(lock);
```

OVERHEAD SUMMARY


Norm. Execution Time