A Halfedge Refinement Rule for Parallel Loop Subdivision

Supplemental Material: CPU Performance Measurements

Kenneth Vanhoey
Unity Technologies

Jonathan Dupuy
Unity Technologies

Performance Measurements. This document provides exhaustive performance measurements of our CPU implementation. The measurements were performed on an AMD Ryzen Threadripper 3960X CPU with 24 cores. We compiled our program with options `-march-native` and `-Os`. Our program relies on `OpenMP` to spawn threads, and we report performance measurements for threads counts 1, 2, 4, 8, 16, and 32. Each number corresponds to the median timing over a set of 50 runs.

Discussion. This document conveys the key information that the performances of our method scale proportionally to the number of CPU threads from 1 to 16 threads. For 32 threads we observe less significant speed-ups, which is due to the fact that the CPU has 24 cores and obviously becomes less efficient at distributing tasks.

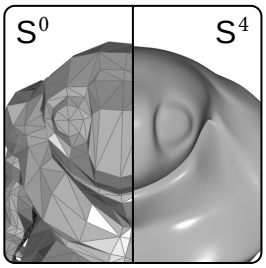
Vertex-Point Refinement. Vertex-point refinement shows an almost proportional increase in performances with respect to thread count.

Halfedge Refinement. Halfedge refinement also shows a proportional trend in general, except for large subdivision depths where the benefits of parallelism attenuates (see, *e.g.*, Sec. 1 at depths six and seven for 8 to 32 threads). We explain this attenuation by the fact that halfedge refinement requires few computations and many memory read/write operations. Thus, as the number of processors increase, memory bandwidth becomes the bottleneck. An interesting avenue of future work would involve optimizations for the memory layout of the halfedge buffer so as to benefit from cache coherency.

Crease Refinement. Crease refinement shows a proportional trend for large subdivision depths. For low subdivision depths, the computation times are negligible even in single-threaded configurations, which sometimes make the graphs look more chaotic.

1 Bigguy

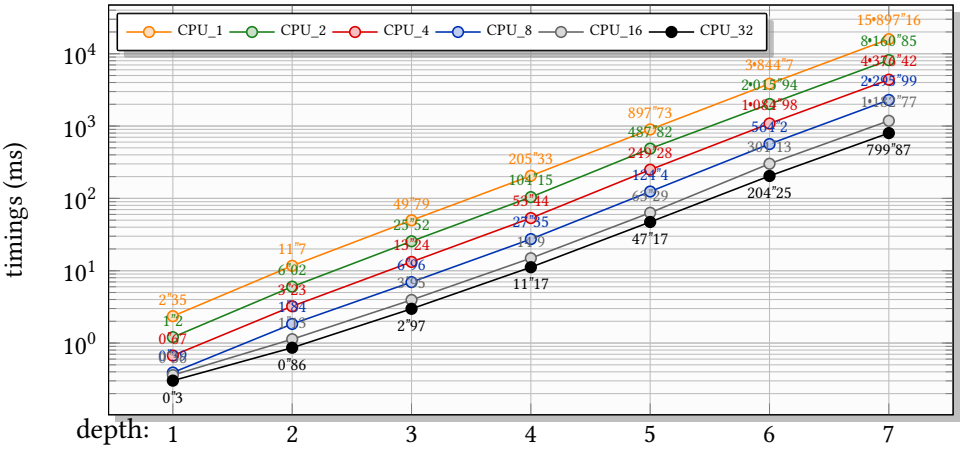
BigguyT



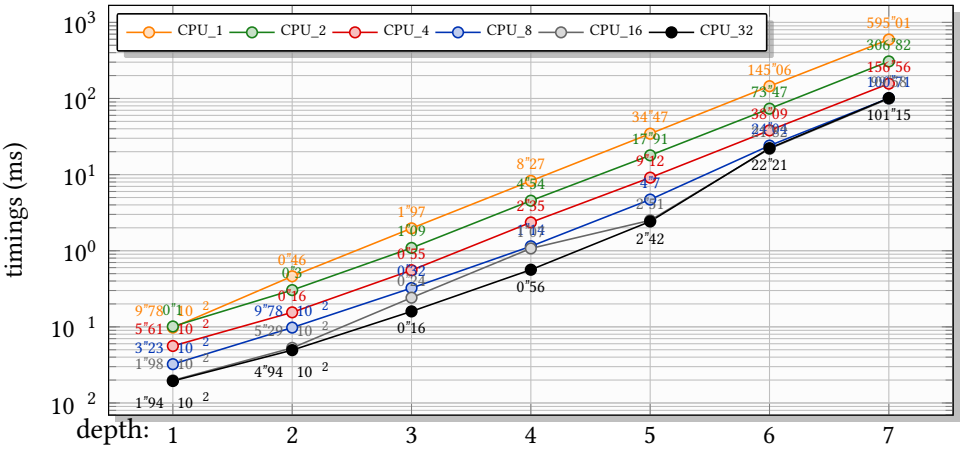
0 boundaries
0 creases

$_0$ = 8•700	$_4$ = 2•227•200
$_0$ = 2•900	$_4$ = 742•400
$_0$ = 4•340	$_4$ = 1•113•600
$+_0$ = 1•452	$+_4$ = 371•202

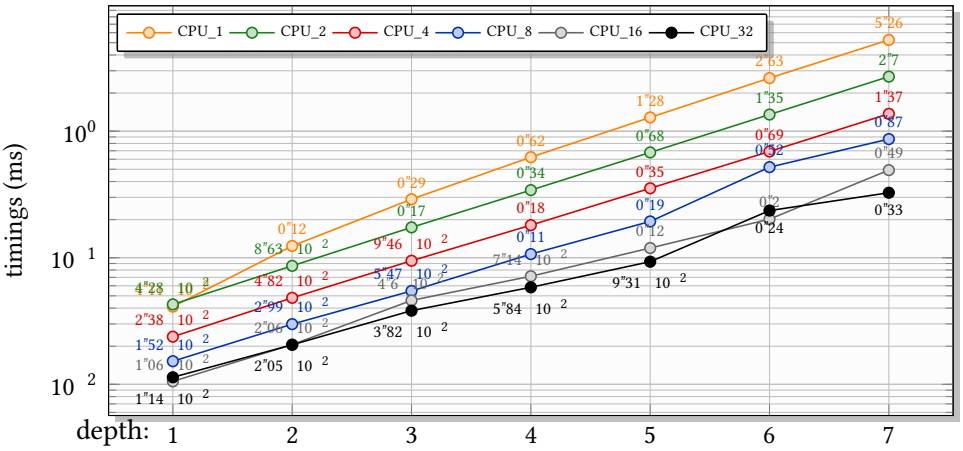
Vertex-Point Refinement



Halfedge Refinement

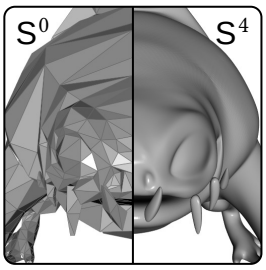


Crease Refinement



2 Monsterfrog

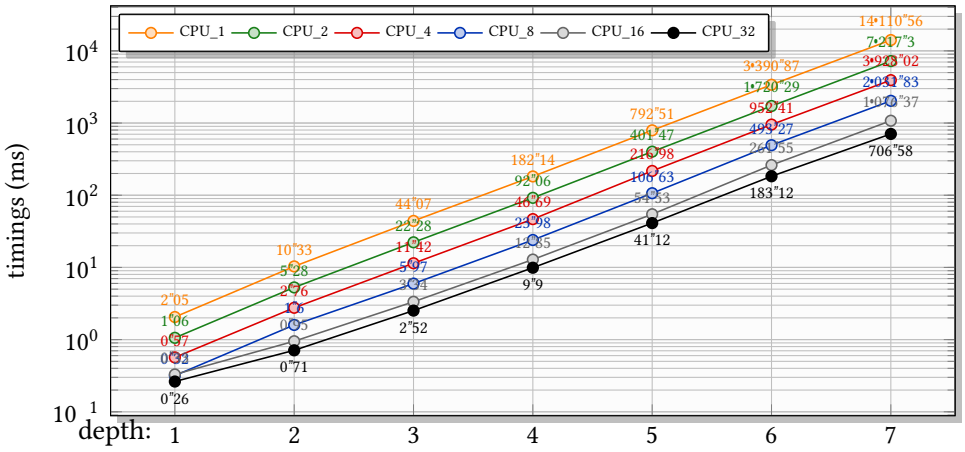
MonsterfrogT



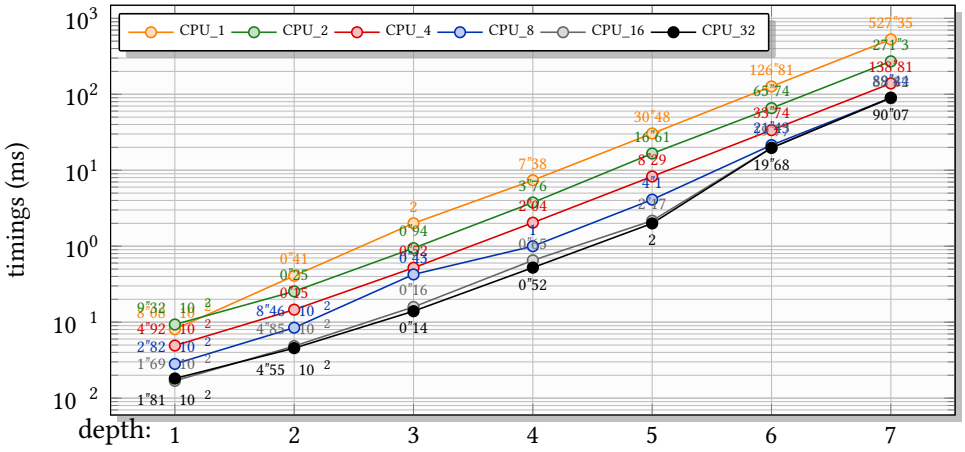
0 boundaries
0 creases

$_0$ = 7 752	$_4$ = 1 984 512
$_0$ = 2 584	$_4$ = 661 504
$_0$ = 3 876	$_4$ = 992 256
$+_0$ = 1 308	$+_4$ = 330 768

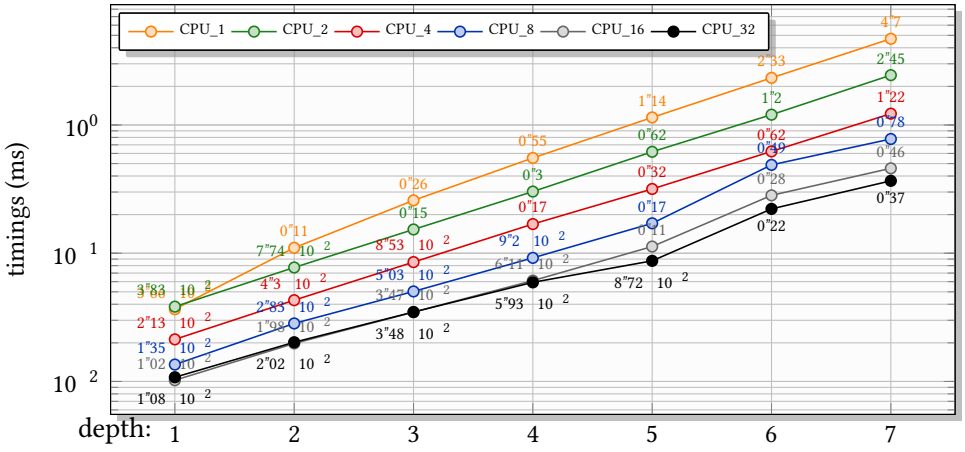
Vertex-Point Refinement



Halfedge Refinement

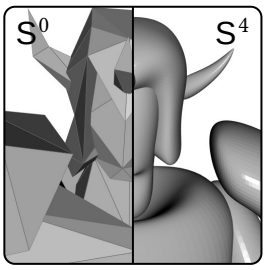


Crease Refinement



3 Knight

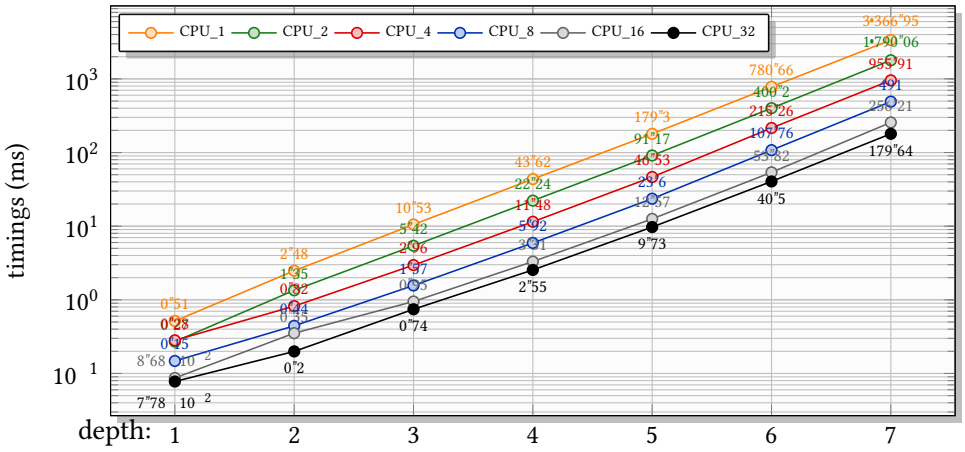
Knight



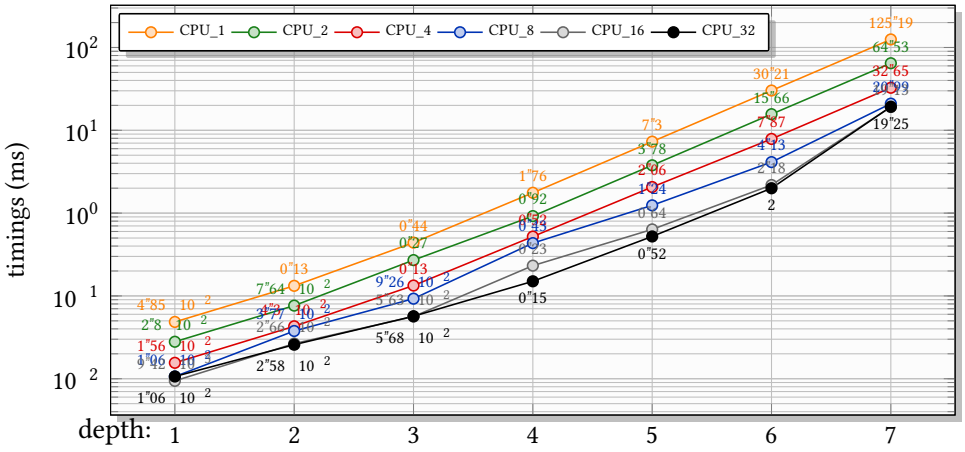
0 boundaries
0 creases

$_0$ = 1•902	$_4$ = 486•912
$_0$ = 634	$_4$ = 162•304
$_0$ = 951	$_4$ = 243•456
$+_0$ = 365	$+_4$ = 81•200

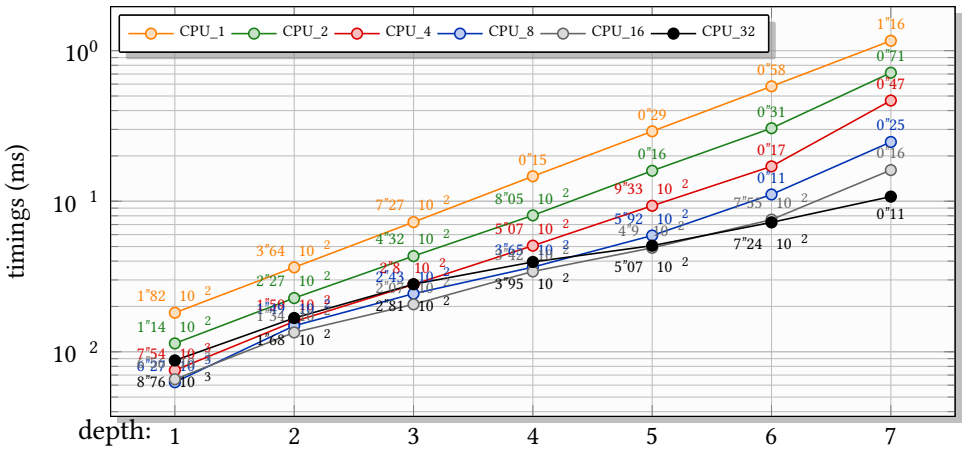
Vertex-Point Refinement



Halfedge Refinement

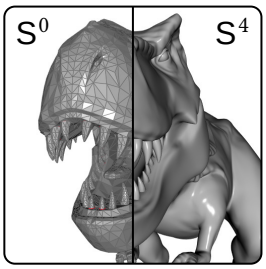


Crease Refinement



4 T-Rex

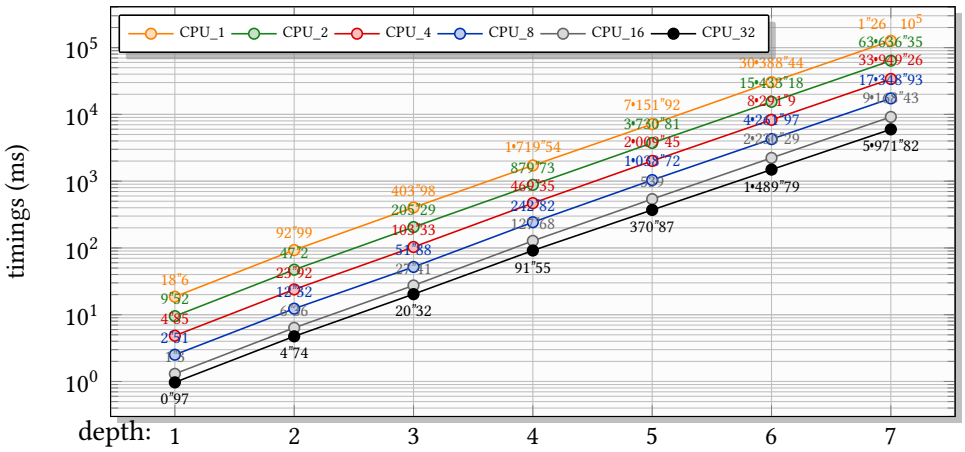
T-rexT



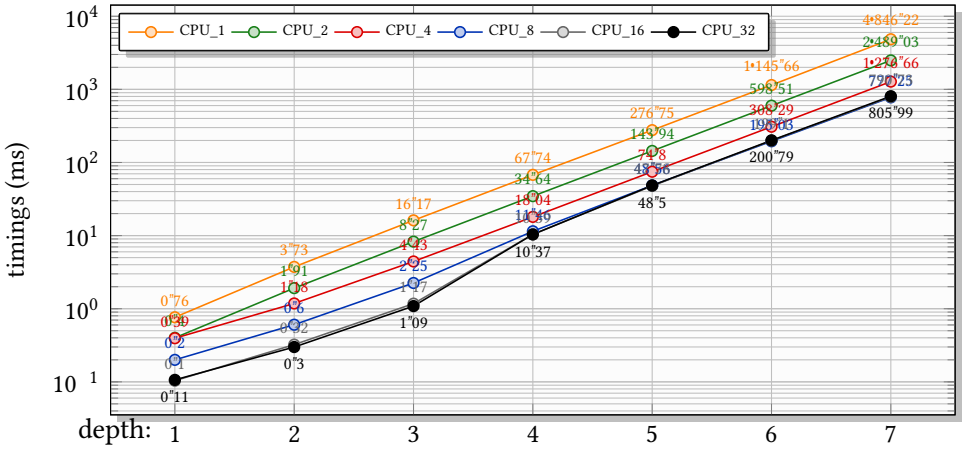
594 boundaries
0 creases

$_0 = 67{,}140$ $_4 = 17{,}187{,}840$
$_0 = 22{,}380$ $_4 = 5{,}729{,}280$
$_0 = 33{,}867$ $_4 = 8{,}598{,}672$
$+_0 = 11{,}539$ $+_4 = 2{,}869{,}444$

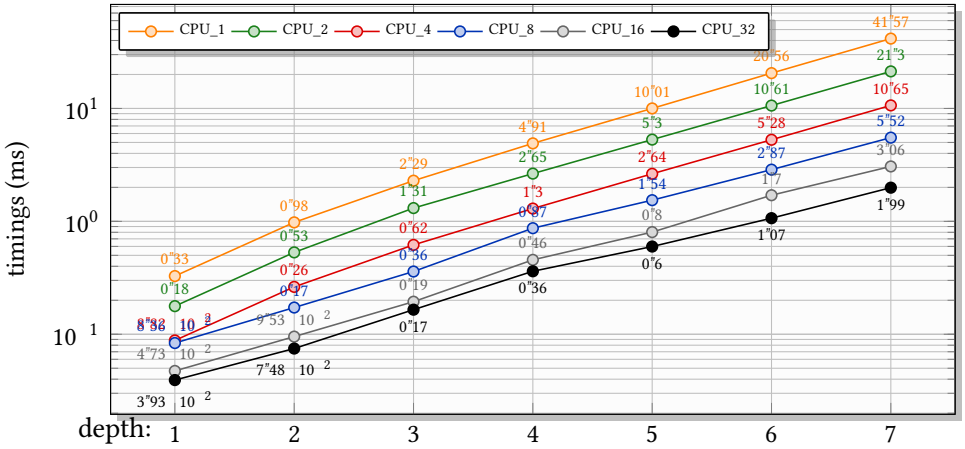
Vertex-Point Refinement



Halfedge Refinement

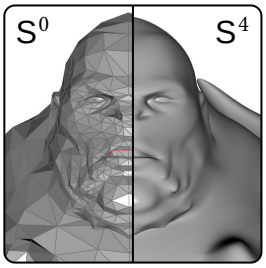


Crease Refinement



5 imrod

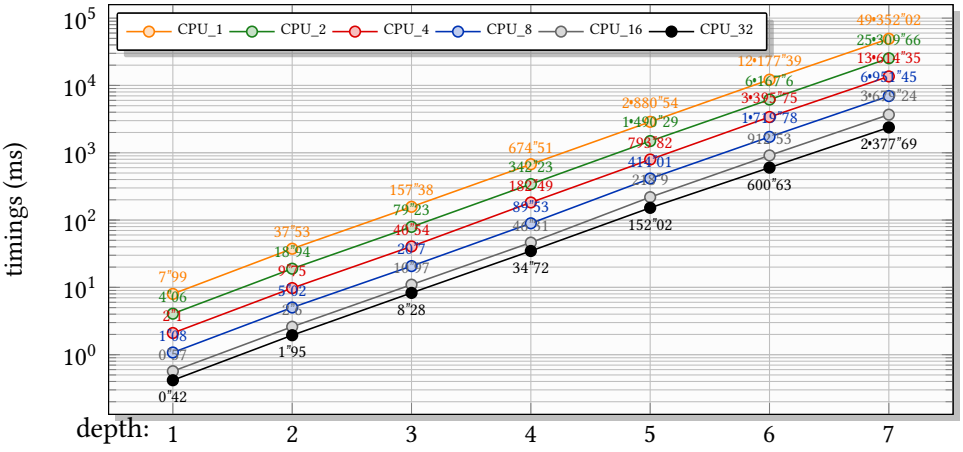
ImrodT



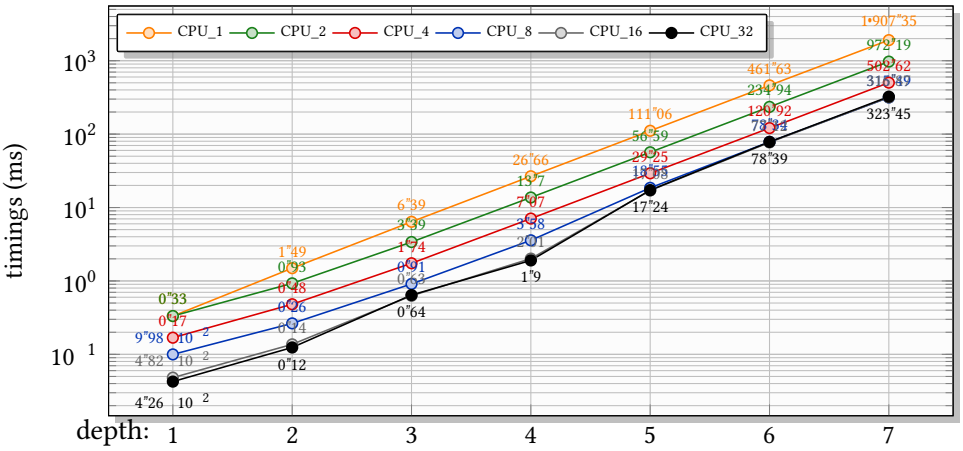
223 boundaries
0 creases

$_0 = 26{,}985$ $_4 = 6{,}908{,}160$
$_0 = 8{,}995$ $_4 = 2{,}302{,}720$
$_0 = 13{,}604$ $_4 = 3{,}455{,}864$
$+_0 = 4{,}630$ $+_4 = 1{,}153{,}165$

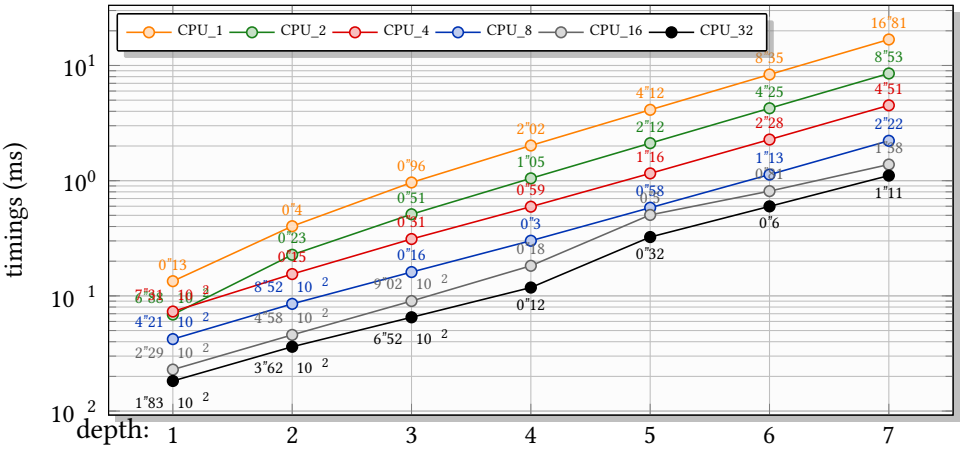
Vertex-Point Refinement



Halfedge Refinement

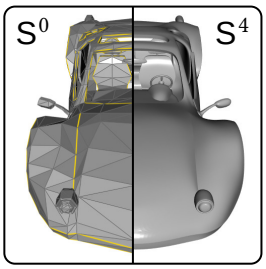


Crease Refinement



6 Car

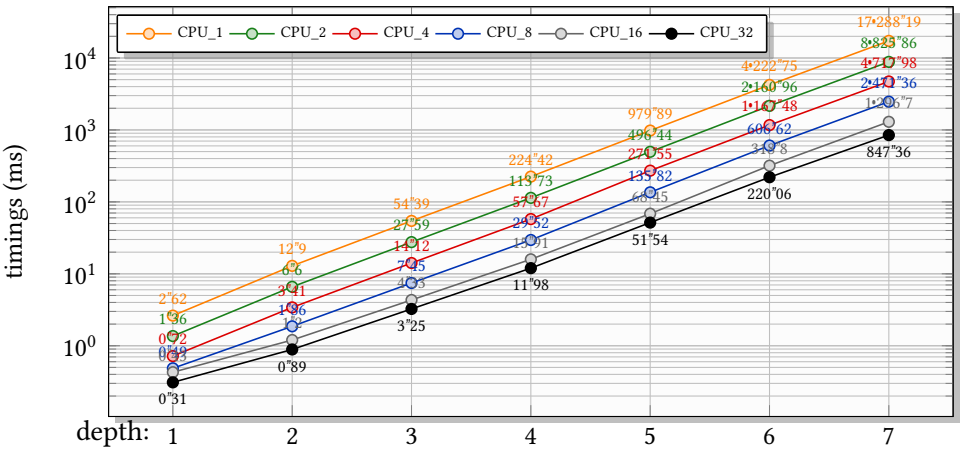
CarT



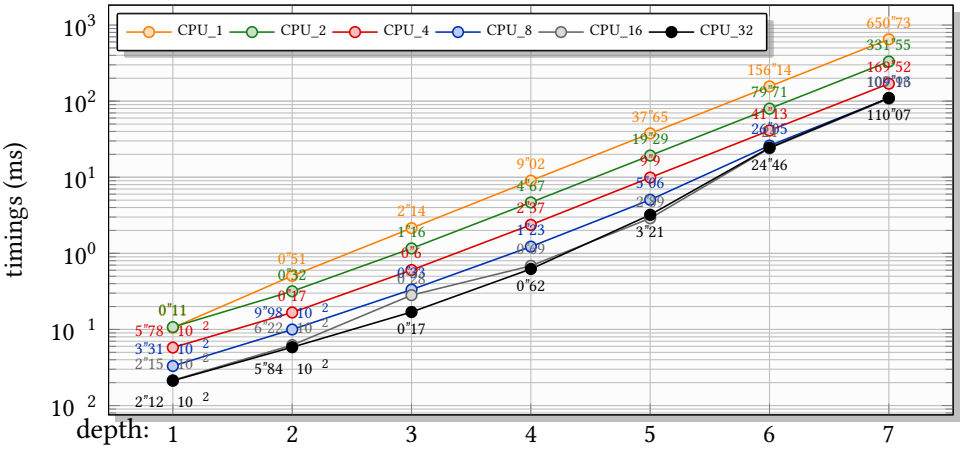
60 boundaries
314 creases

$_0 = 9{,}450$ $_4 = 2{,}419{,}200$
$_0 = 3{,}150$ $_4 = 806{,}400$
$_0 = 4{,}755$ $_4 = 1{,}210{,}080$
$+_0 = 1{,}642$ $+_4 = 403{,}717$

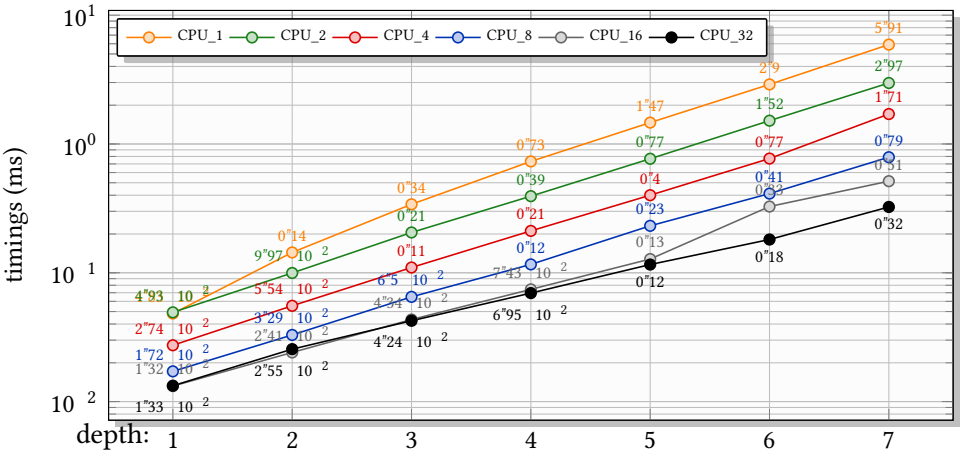
Vertex-Point Refinement



Halfedge Refinement

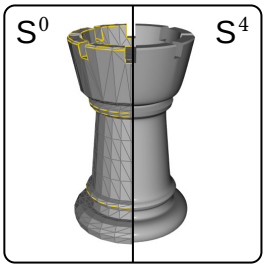


Crease Refinement



7 Rook

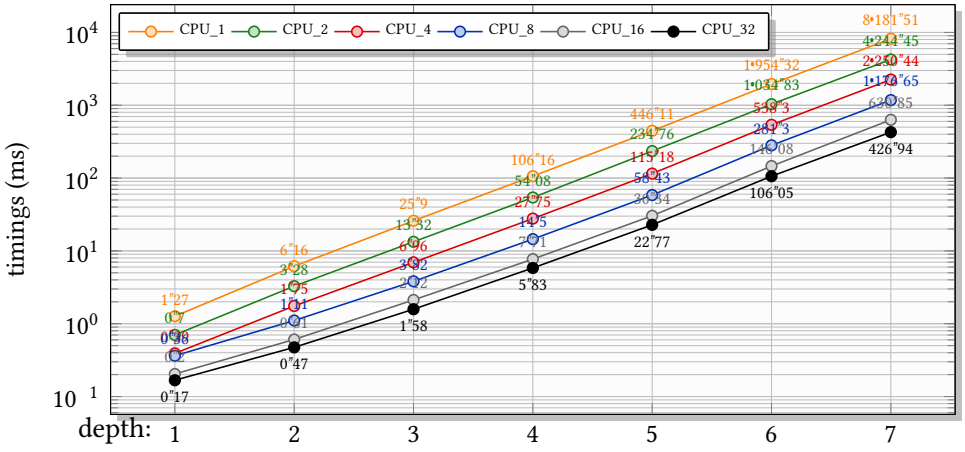
RookT



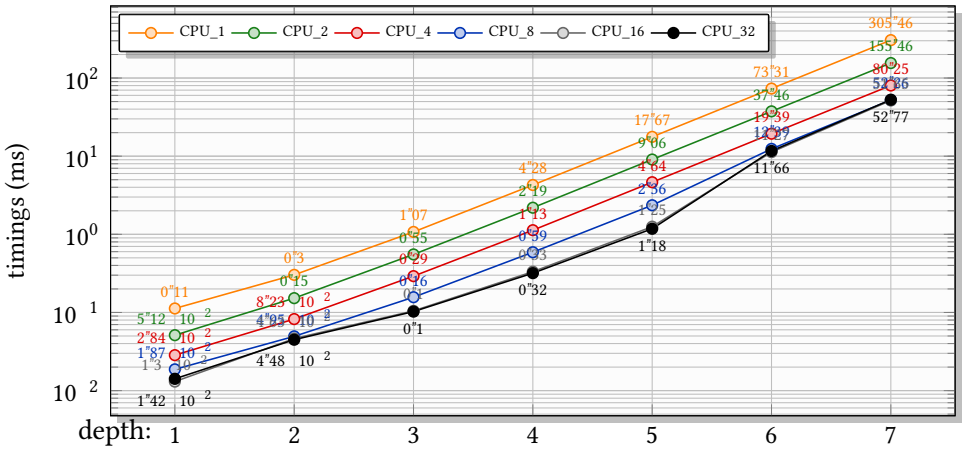
24 boundaries
280 creases

$_0$ = 4• 530	$_4$ = 1• 159• 680
$_0$ = 1• 510	$_4$ = 386• 560
$_0$ = 2• 277	$_4$ = 580• 032
$+_0$ = 768	$+_4$ = 193• 473

Vertex-Point Refinement



Halfedge Refinement



Crease Refinement