Need to Node: Profiling Node.js Applications Patrick Mueller





Questions during the Need to Node webinar?

Post a question to Twitter with the hashtag:

#needtonode





NodeSource is *the* Enterprise Node.js company offering a suite of products and professional services explicitly focused on the needs of Enterprise users of Node.js.





NODESOURCE TM



Agenda

Introduction to profiling Node.js applications

- of profiling is available for Node.js
- applications
- Demonstrate using the Node.js profilers •



Explain what profiling applications means, and what kind

Show what insights are provided when profiling Node.js





What is profiling?



What is profiling?

Profiling is:

- capturing statistics while a program is running •
- displaying those statistics with useful visualizations ●

Provides deep view into application performance no more guessing at what your program is doing •





What kind of profiling can I do with Node.js?

Profiling Node.js applications

- **V8 heap snapshot profiler** measures memory usage •
 - find memory leaks •
 - optimize memory use by your program •
- **V8 CPU profiler** measures function execution time •
 - find bottlenecks in your application
 - optimize the performance of your program ●









What is profiling?

What is the real value in profiling?

- Run your programs faster •
- Run your programs with less RAM •
- code bottlenecks

Save \$\$\$!!!



Find difficult to diagnose problems - leaking memory and



What kind of insights does the heap snapshot profiler provide?





Heap Snapshot profiler

What does the heap snapshot profiler do?

- between those objects.
 - your application.



While your program is running, generates a JSON-able description of all JavaScript objects allocated in your program at a specific point in time, and the references

This description is quite large; plan on 2x RAM usage of

 Visualization shows object counts/sizes by constructor (class), and references to and from individual objects.



Heap Snapshot insights - counts/sizes per class (N|Solid)

Constructors				
FILTER e.g. Array				
Point2D				
(system)				
▶ (array)				
(string)				
(compiled code)				
(closure)				
Object				



done imes

RETAINED SIZE	SHALLOW SIZE	OBJECTS V
1.91 MB	1.91 MB	41710
588.29 KB	426.81 KB	9855
4.63 MB	2.52 MB	5450
1.37 MB	1.37 MB	4719
2.41 MB	1.24 MB	4120
1.4 MB	163.62 KB	2327
7.72 MB	42.97 KB	829



Heap Snapshot insights

- shallow size vs retained size
 - **shallow size** amount of memory this object uses just by itself; typically only relevant for Strings and Arrays
 - retained size total amount of memory this object is referencing that would be garbage collected (GC'd) if this object was garbage collected - usually the more interesting number
- Heap Snapshots are typically grouped by constructor name, so literal objects are all lumped into Object





Heap Snapshot insights - snapshot diff (Chrome Dev Tools)

Comparison V Class filter mem-haw	g-1 <mark>-89b2</mark>	131d6	7b5c1ca143	23	57c76e	b0abd1b2e5f	7c-1453125	680441 🔻
Constructor	# Nev	v	# Deleted	#	Delta	Alloc. Size▼	Freed Size	Size Delta
▶ Point2D		1156	0		+1156	55 488	0	+55 488
▶ (compiled code)		1/0	598		-420	34 112	352 128	-318016
▶ (array)		264	821		-557	19 368	81 784	-62 416
▶ (system)		97	626		-529	2 3 2 8	17 552	-15 224
▶ (string)		1	185		-184	40	7 688	-7 648
▶ Array		1	1		C	32	32	0
▶ (concatenated string)		0	23		-23	0	920	-920
Retainers								=
Object			Distan	ce	🔺 Sh	allow Size	Retained	d Size
From the time the first sn snapshot was taken, 1	apsh 156 r	not new	was ta / Poin	al t2	ken, 2D c	, till th objects	e seco s were	ond



created, and none were garbage collected



Heap Snapshot insights - which objects reference selected object (Chrome Dev Tools)



14 © 2016 NodeSource Confidential

(N|S)

	Distance	Distance		Objects Count			Retained Size	
		-	6 177	8%	3 0 5 5 7 8 4	35%	5 294 736	61
		3	191	0 %	20 432	0 %	3 957 092	45
		5	4	0 %	128	0 %	3 837 664	44
		3	4 394	6%	1 4 1 2 6 4 0	16%	2 635 360	30
		7	41710	57%	2 002 056	23%	2 002 056	23
		11			48	0%	48	C
		11			48	0 %	48	0
		11			48	0 %	48	Q
			Ini	S P(oint2D	ODJ	ect is	
			In k by	s Po pein y th noo	oint2D Ig refer is Leak dule va	obj end yCa ria	ect is ced ache ble	
			In k by	s Po pein y th noo	oint2D Int2D Is refer Is Leak dule va	obj ene yCa ria	ect is ced ache ble	0
			In k by	S Po pein y th noo ² 1	oint2D ng refer is Leak dule va	obj end yCa ria	ect is ced ache ble	0
31			In k by	S Po pein th noo 2 1 3	oint2D Int2D Int2D Is refer Is Leak dule va	obj end yCa ria	ect is ced ache ble	
31			Ini k by	SPO pein th noo 2 1 3 4	oint2D Int2D Int2D Is Leak dule va	obj end yCa ria	Pect IS ced ache ble 72 9 344 32 56	



What kind of insights does CPU profiling provide?





CPU profiler

What does the CPU profiler do?

- profiler.

 - a JSON-able object.
- execution time, and aggregate function call times.



While your program is running, you can start the profiler, let it run for some number of seconds, and then stop the

• While the profiler is running, it collects the stack of functions being executed, at a sub-millisecond interval.

• When the profiler is stopped, that data is aggregated into

Visualizations show object function call stacks, function



CPU profiling insights - flame graph (N|Solid)





done 🗙

CALLS	FUNCTION	SELF	TOTAL
89	doStuff	112ms	112ms
	samples/cpu-profile.js	17.45%	17.45%
	x	2ms	114ms
	samples/cpu-profile.js	0.20%	17.65%
		Ome	311mc
0	y samples/cnu-profile is	0.00%	18 130%
		0.00%	40.4370
	Z	2ms	485ms
	samples/cpu-profile.js	0.20%	75.69%
0	main	0ms	630ms
	samples/cpu-profile.js	0.00%	98.24%
	listOnTimeout	Oms	630ms
0	timore is	0.0004	09 2404
	umers.js	0.00%	98.24%



CPU profiling insights

- total time vs self time
 - **total time** total elapsed time spent in a function
 - **self time** -total elapsed time spent in a function, minus total time spent in functions called from this function
- **source location** of functions provided in profiling data •
- name your functions, lest you see lots of functions named ● "(anonymous function)"





CPU profiling insights - sunburst (N|Solid)





done 🗙

CALLS	FUNCTION	SELF	TOTAL
158	doStuff samples/cpu-hawg.j s	198ms 30.92%	198ms 30.92%
2	y samples/cpu-hawg.j s	3ms 0.39%	309ms 48.34%
1	z samples/cpu-hawg.j s	2ms 0.20%	480ms 74.95%
0	processThings samples/cpu-hawg.j s	0ms 0.00%	622ms 97.26%
0	wrapper timers.js	0ms 0.00%	622ms 97.26%
2	listOnTimeout timers.js	3ms 0.39%	625ms 97.65%



CPU profiling insights - treemap (N|Solid)





done 🗙

ALLS	FUNCTION	SELF	TOTAL
158	doStuff samples/cpu-hawg.j s	198ms 30.92%	198ms 30.92%
2	y samples/cpu-hawg.j s	3ms 0.39%	309ms 48.34%
1	z samples/cpu-hawg.j s	2ms 0.20%	480ms 74.95%
0	processThings samples/cpu-hawg.j s	0ms 0.00%	622ms 97.26%
0	wrapper timers.js	0ms 0.00%	622ms 97.26%
2	listOnTimeout timers.js	3ms 0.39%	625ms 97.65%



CPU profiling insights- tablular view of functions (Chrome Dev Tools)

Heavy (Bot	tom Up)	• • ×	Ç	
Self		Tota	▼ 1	Function
4364.6 ms		4364.6 ms		(idle)
0 ms	0 %	630.0 ms	98.24%	listOnTimeout
0 ms	0 %	630.0 ms	98.24%	▶ main
626.2 ms	97.65%	626.2 ms	97.65%	►doStuff
1.3 ms	0.20%	485.4 ms	75.69%	►z
0 ms	0 %	310.6 ms	48.43%	▶y
0 ms	0 %	144.6 ms	22.55%	▶a
0 ms	0 %	137.1 ms	21.37%	▶b
0 ms	0 %	122.0 ms	19.02%	►c
1.3 ms	0.20%	113.2 ms	17.65%	►x
0 ms	0 %	96.8 ms	15.10%	►d
0 ms	0 %	66.6 ms	10.39%	►e
1.3 ms	0.20%	25.1 ms	3.92%	▶f
5.0 ms	0.78%	5.0 ms	0.78%	(program)
2.5 ms	0.39%	2.5 ms	0.39%	(garbage collector)
0 ms	0 %	1.3 ms	0.20%	▶ readableAddChunk
0 ms	0%	1.3 ms	0.20%	processStats
0 ms	0 %	1.3 ms	0.20%	► ChildProcess.spawn
0 ms	0 %	1.3 ms	0.20%	▶exports.stat
1.3 ms	0.20%	1.3 ms	0.20%	▶ spawn
1.3 ms	0.20%	1.3 ms	0.20%	▶ stopProfiling
0 ms	0 %	1.3 ms	0.20%	▶exports.spawn
1.3 ms	0.20%	1.3 ms	0.20%	▶ Readable.read
0 ms	0 %	1.3 ms	0.20%	▶ exports.execFile
0 ms	0 %	1.3 ms	0.20%	▶exports.exec
0 ms	0 %	1.3 ms	0.20%	▶ Readable.push
0 ms	0 %	1.3 ms	0.20%	start
0 ms	0 %	1.3 ms	0.20%	profiler.startProfiling
0 ms	0 %	1.3 ms	0.20%	▶ stats.ps
0 ms	0 %	1.3 ms	0.20%	► (anonymous function)
0 ms	0 %	1.3 ms	0.20%	onread



(program):-1 timers.js:55 /Users/pmuellr/Projects/slides/2016/01-intro-to-profiling/samples/cpu-profile.js:14 /Users/pmuellr/Projects/slides/2016/01-intro-to-profiling/samples/cpu-profile.js:30 /Users/pmuellr/Projects/slides/2016/01-intro-to-profiling/samples/cpu-profile.js:26 /Users/pmuellr/Projects/slides/2016/01-intro-to-profiling/samples/cpu-profile.js:27 /Users/pmuellr/Projects/slides/2016/01-intro-to-profiling/samples/cpu-profile.js:19 /Users/pmuellr/Projects/slides/2016/01-intro-to-profiling/samples/cpu-profile.js:20 /Users/pmuellr/Projects/slides/2016/01-intro-to-profiling/samples/cpu-profile.js:21 /Users/pmuellr/Projects/slides/2016/01-intro-to-profiling/samples/cpu-profile.js:28 /Users/pmuellr/Projects/slides/2016/01-intro-to-profiling/samples/cpu-profile.js:22 /Users/pmuellr/Projects/slides/2016/01-intro-to-profiling/samples/cpu-profile.js:23 /Users/pmuellr/Projects/slides/2016/01-intro-to-profiling/samples/cpu-profile.js:24 (program):-1 (program):-1 _stream_readable.js:122 nsolid.js:247 internal/child_process.js:251 nsolid.js:1165 (program):-1 (program):-1 child_process.js:355 _stream_readable.js:249 child_process.js:117 child_process.js:108 _stream_readable.js:98 nsolid.js:105 profiler.js:174 nsolid.js:1315 nsolid.js:1136 <u>net.js:500</u>



How do you get profiling information from Node.js applications?





Profiling tools

v8-profiler package on npm

• into Chrome Dev Tools

N Solid from NodeSource

•



manually instrument your application, load profile data

generate and display profiles at the click of a button



Profiling with v8-profiler

using v8-profiler from npm

- npm install v8-profiler
- add v8-profiler to your package.json dependencies, if not • running locally
- add code to your app for triggers for starting/stopping CPU profiles, and to generate heap snapshots
 - the triggers will:
 - call functions in v8-profiler
 - save results of v8-profiler function calls into JSON files •
- run your app and trigger the profiles you want to generate ● load JSON files into the Profiles tab of Chrome Dev Tools •



Profiling with NodeSource's N|Solid

what is N Solid?

- has been enhanced to provide additional runtime diagnostics.
- Provides a web-based console to:
 - monitor applications at scale, in production •
 - obtain process- and system-specific statistics for an • individual application instance
 - obtain and view CPU profiles and heap snapshots at the • click of a button

• NSolid is a fully compatible Node.js v4.x LTS runtime that



Profiling with NodeSource's N Solid

using N Solid

- Run your app with the N Solid runtime ●
- heap snapshot
- Tools



Open the N Solid Console in a browser, navigate to your application, press the buttons to generate a CPU profile or

The results are shown in the N|Solid Console, and the data is available to download for further analysis in Chrome Dev





27 © 2016 NodeSource Confidential

Demotime!

sample applications being profiled: https://gist.github.com/pmuellr/2c7e9c7b95352d1b33e0



Get involved in building new profiling tools!

follow the Node.js Tracing Work Group

https://github.com/nodejs/tracing-wg/issues/38

write your own profiling visualizers

CPU profiles and heap snapshots are just JSON! •







N Solid references

- **Download N Solid free for development**: • https://nodesource.com/products/nsolid
- **N**|Solid documentation: • https://docs.nodesource.com/
- Getting Started with N|Solid: •
- Getting Started with the N Solid Console: • https://nodesource.com/blog/getting-started-with-the-nsolid-console/



https://nodesource.com/blog/getting-started-with-nsolid-at-the-command-line/



V8 profiling references

- snapshots: problems/heap-snapshots
- profiles: <u>is-execution</u>
- https://www.npmjs.com/package/v8-profiler



Google Developers - "How to Record Heap Snapshots" - introduction to heap

<u>https://developers.google.com/web/tools/chrome-devtools/profile/memory-</u>

Google Developers - "Speed Up JavaScript Execution" - introduction to CPU

<u>https://developers.google.com/web/tools/chrome-devtools/profile/rendering-tools/</u>

v8-profiler package at npm - open source package exposing V8's profiling APIs:



Questions during the Need to Node webcast?

post a question to Twitter with the hashtag

#needtonode



(N|S) 31 © 2016 NodeSource Confidential

these slides at <u>pmuellr.github.io/slides</u>



Thank you.

Patrick Mueller pmuellr@nodesource.com

@pmuellr





