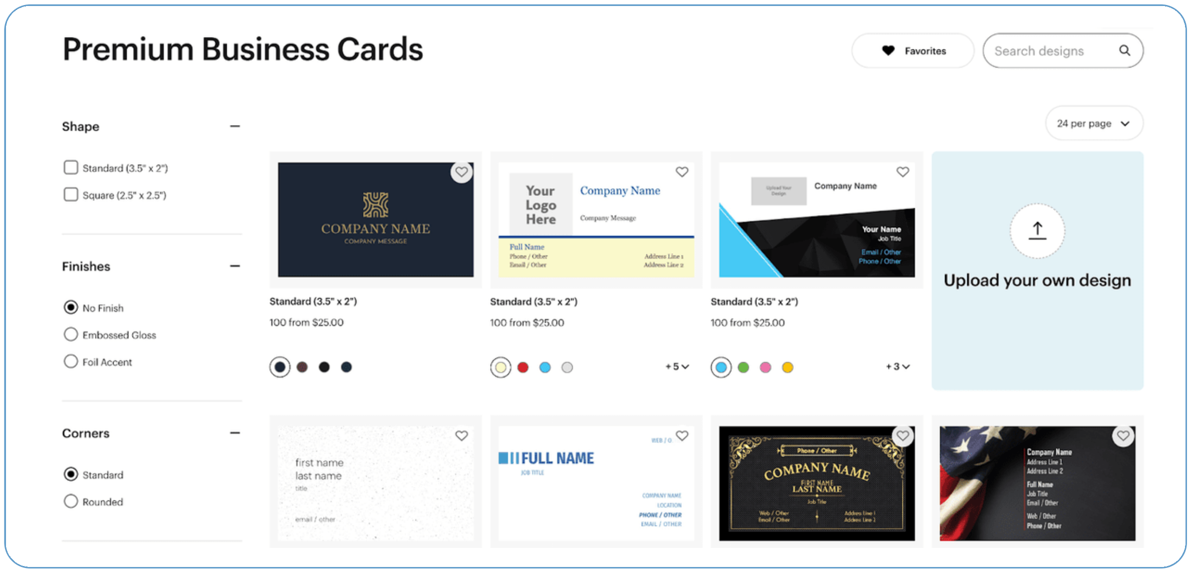
SPEEDSCALE

CASE STUDY

Cimpress reduces load testing time by 80% with Speedscale

Cimpress is a global eCommerce company that produces custom marketing materials for small businesses. Millions of customers worldwide use their design and printing services to create business cards, signage, promotional items, stationary, gifts, and more.



Premium Business Cards

Favorites | Search designs

24 per page

Shape
- ☐ Standard (3.5" x 2")
- ☐ Square (2.5" x 2.5")

Finishes
- [✓ No Finish]
- Embossed Gloss
- Foil Accent

Corners
- [✓ Standard]
- Rounded

Standard (3.5" x 2")
100 from $25.00

Standard (3.5" x 2")
100 from $25.00
+5

Standard (3.5" x 2")
100 from $25.00
+3

Upload your own design

Challenge

As Principal Software Engineer, Jeff Kwan leads a team of API engineers that manages the software and product data behind Cimpress' expansive catalog. "The fidelity of this product data is very important because if it's wrong, then a lot of downstream activities are impacted—product design, fulfillment, prices. There's a lot that could go wrong if the software isn't working the right way," Jeff says.

Jeff's team had inherited legacy software from a different team and was looking to migrate off of it. He realized they needed a solution to test that the new software worked the same way as the old one. To describe the challenge he was facing, Jeff explains:

> Today, I know my endpoint for this ID produces this result. Tomorrow, when I make a change in the code, I need to know, 'Did I get the same result?' It sounds simple, but there's actually a lot of complexity behind answering this question confidently.

Jeff Kwan
Principal Software Engineer

cimpress

Evaluating Solutions

Before the team found Speedscale, they had tried a couple open source tools, Diffy and GoReplay. While the tools initially met their needs, they soon felt that certain key elements were missing, like clear documentation, a thoughtful developer experience, and customer support.

> As an engineering leader, it was hard for me to tell my team to use a tool that only partially solved our problem, leaving them with the challenge of figuring out how to make it work. What we needed was a more complete API testing solution long term.

For about two years, Jeff and his team created workarounds with the open source tooling, but were still struggling with the lack of a more comprehensive solution. His confidence level in executing tasks, like changing framework versions, was not as high as he would have liked it to be.

Jeff came across Speedscale through a Google search while looking for automated API testing tools, and recalls it was the pure focus on APIs and Kubernetes, as well as good documentation that initially attracted him to the offering. Upon further investigation, he grew more impressed by the Speedscale team's technical expertise and the product's capabilities. Right off the bat, Jeff understood one of the key values of Speedscale's offering: speed of implementation.

"When I would try recording traffic on my own, it was always a pain. I thought to myself, if Speedscale can record traffic and then replay it, that means, at a minimum, setup will be easier."

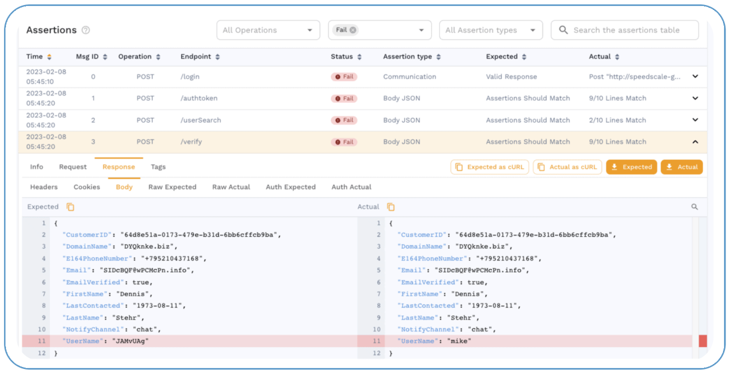
Using Speedscale

Jeff recalls a very smooth implementation, guided mostly by documentation, a few calls with the Speedscale team, and engagement with other developers in Speedscale's Slack community.

Today, Jeff's team uses Speedscale for 3 main use cases: Regression Testing, Load Testing, and Observability.

Regression Testing

Despite having talented, senior engineers on the team, catching regression bugs was still difficult without a solution like Speedscale. Data setup and scripting was time-consuming, and the overall testing process was inconsistent, involving light spot-checking instead of a more comprehensive approach to testing. Now, they can record and apply real customer traffic to validate that things like platform or node version updates aren't changing the outputs.



Time	Msg ID	Operation	Endpoint	Status	Assertion type	Expected	Actual
2023-02-08 05:45:10	0	POST	/login	Fail	Communication	Valid Response	Post "http://speedscale-g...
2023-02-08 05:45:20	1	POST	/authtoken	Fail	Body JSON	Assertions Should Match	9/10 Lines Match
2023-02-08 05:45:20	2	POST	/userSearch	Fail	Body JSON	Assertions Should Match	2/10 Lines Match
2023-02-08 05:45:20	3	POST	/verify	Fail	Body JSON	Assertions Should Match	9/10 Lines Match

Expected

```
{
  "CustomerID": "64d8e51a-0173-479e-b31d-6bb6cffcb9ba",
  "DomainName": "DYQknke.biz",
  "E164PhoneNumber": "+795210437168",
  "Email": "SIDcBQF@wPCMcPn.info",
  "EmailVerified": true,
  "FirstName": "Dennis",
  "LastContacted": "1973-08-11",
  "LastName": "Stehr",
  "NotifyChannel": "chat",
  "UserName": "JAMvUAg"
}
```

Actual

```
{
  "CustomerID": "64d8e51a-0173-479e-b31d-6bb6cffcb9ba",
  "DomainName": "DYQknke.biz",
  "E164PhoneNumber": "+795210437168",
  "Email": "SIDcBQF@wPCMcPn.info",
  "EmailVerified": true,
  "FirstName": "Dennis",
  "LastContacted": "1973-08-11",
  "LastName": "Stehr",
  "NotifyChannel": "chat",
  "UserName": "mike"
}
```

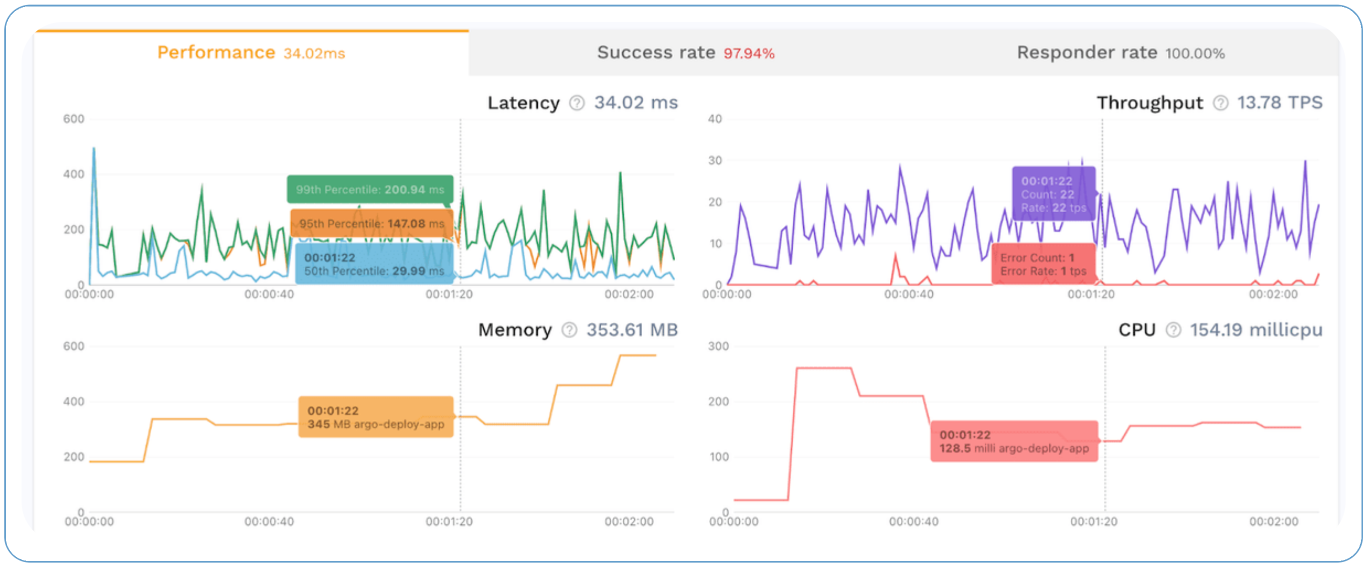
Load Testing

Prior to Speedscale, Jeff and his team had attempted to test application performance by scripting tests in K6. But this had 2 major drawbacks: 1) it was a manual process, and 2) scripting tests with the wrong product data could stress parts of the product catalog that weren't realistic. Then, Jeff discovered Speedscale's [K6 converter]() and was able to record real customer traffic and run it through K6, taking a 3-4 day testing process for one service down to 40 minutes. Today, Jeff uses Speedscale's own pod-based load generators to bypass K6 altogether, saving even more time.

But what he really liked about Speedscale was that it gave him more practical data.

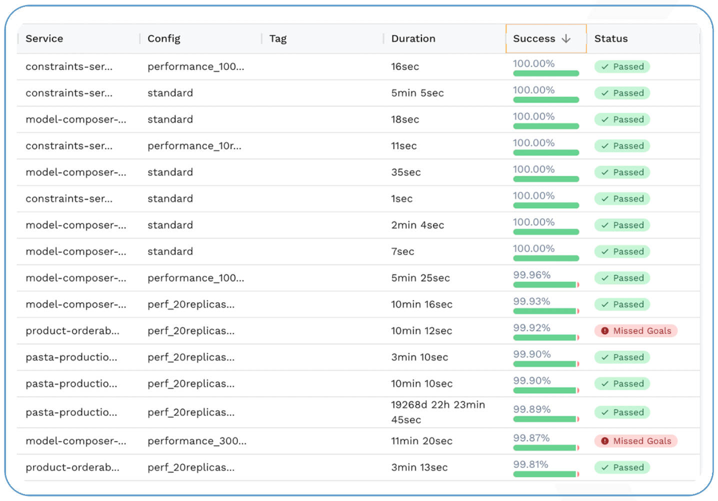
> Speedscale gave us clear insight on the upper load limit of our software. Whereas with K6, it was more binary, like a PASS/FAIL grade. Speedscale took it one step further to help us understand, based on transactions per second, exactly where our breakpoints were. We felt more confident in where our service could be.

As Jeff and his team prepared for Black Friday, they reduced a typical 4-week testing process down to 3-4 days with Speedscale.



API Observability

Since implementing Speedscale, Jeff and his team have realized additional benefits beyond their initial expectations. Speedscale's capabilities around API observability have helped them pinpoint and fix specific issues that they wouldn't have discovered otherwise.



Service	Config	Tag	Duration	Success ↓	Status
constraints-ser...	performance_100...		16sec	100.00%	✓ Passed
constraints-ser...	standard		5min 5sec	100.00%	✓ Passed
model-composer-...	standard		18sec	100.00%	✓ Passed
constraints-ser...	performance_10r...		11sec	100.00%	✓ Passed
model-composer-...	standard		35sec	100.00%	✓ Passed
constraints-ser...	standard		1sec	100.00%	✓ Passed
model-composer-...	standard		2min 4sec	100.00%	✓ Passed
model-composer-...	standard		7sec	100.00%	✓ Passed
model-composer-...	performance_100...		5min 25sec	99.96%	✓ Passed
model-composer-...	perf_20replicas...		10min 16sec	99.93%	✓ Passed
product-orderab...	perf_20replicas...		10min 12sec	99.92%	● Missed Goals
pasta-productio...	perf_20replicas...		3min 10sec	99.90%	✓ Passed
pasta-productio...	perf_20replicas...		10min 10sec	99.90%	✓ Passed
pasta-productio...	perf_20replicas...		19268d 22h 23min 45sec	99.89%	✓ Passed
model-composer-...	performance_300...		11min 20sec	99.87%	● Missed Goals
product-orderab...	perf_20replicas...		3min 13sec	99.81%	✓ Passed

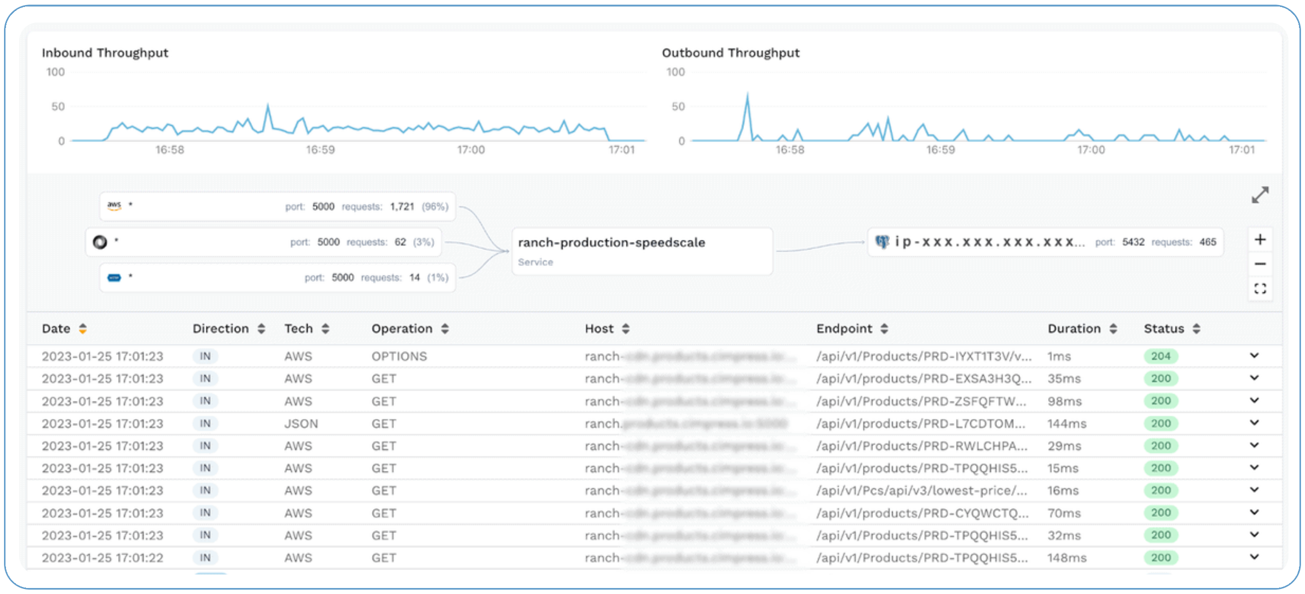
"In one instance, I noticed that there was a small amount of traffic that had been causing some failures in our system. It was only happening 3% of the time, but it was still somehow critical to how it all worked. So, I went in and fixed it, but I wouldn't have found that issue had I not seen it in Speedscale." He was also able to identify when calls weren't optimized properly, making direct calls to resources as opposed to using caches. Optimizations such as those could all add up during holiday load.

Results & Future Outlook

One year later, Jeff and his team report that they are incredibly pleased with their experience and the results they've achieved with Speedscale.

"Speedscale's offering is really, really good. The products, especially Traffic Viewer and Diff Viewer, and the overall developer-centric experience are all very polished.

He appreciates how open the Speedscale team is to receiving product feedback, and is excited to have his team trial some new features, like local desktop testing and auto-updating of web tokens.


Inbound Throughput
100
50
0
16:58
16:59
17:00
17:01
Outbound Throughput
100
50
0
16:58
16:59
17:00
17:01
aws * port: 5000 requests: 1,721 (96%)
* port: 5000 requests: 62 (3%)
* port: 5000 requests: 14 (1%)
ranch-production-speedscale
Service
ip-xxx.xxx.xxx.xxx... port: 5432 requests: 465

Date	Direction	Tech	Operation	Host	Endpoint	Duration	Status
2023-01-25 17:01:23	IN	AWS	OPTIONS	ranch-[redacted]	/api/v1/Products/PRD-IYXT1T3V/v...	1ms	204
2023-01-25 17:01:23	IN	AWS	GET	ranch-[redacted]	/api/v1/products/PRD-EXSA3H3Q...	35ms	200
2023-01-25 17:01:23	IN	AWS	GET	ranch-[redacted]	/api/v1/Products/PRD-ZSFQFTW...	98ms	200
2023-01-25 17:01:23	IN	JSON	GET	ranch.[redacted]	/api/v1/products/PRD-L7CDTOM...	144ms	200
2023-01-25 17:01:23	IN	AWS	GET	ranch-[redacted]	/api/v1/Products/PRD-RWLCHPA...	29ms	200
2023-01-25 17:01:23	IN	AWS	GET	ranch-[redacted]	/api/v1/products/PRD-TPQQHIS5...	15ms	200
2023-01-25 17:01:23	IN	AWS	GET	ranch-[redacted]	/api/v1/Pcs/api/v3/lowest-price/...	16ms	200
2023-01-25 17:01:23	IN	AWS	GET	ranch-[redacted]	/api/v1/products/PRD-CYQWCTQ...	70ms	200
2023-01-25 17:01:23	IN	AWS	GET	ranch-[redacted]	/api/v1/Products/PRD-TPQQHIS5...	32ms	200
2023-01-25 17:01:22	IN	AWS	GET	ranch-[redacted]	/api/v1/Products/PRD-TPQQHIS5...	148ms	200