

TECH & AI CASE STUDY - MEDIA & EVENTS - 2026

Serverless Pipeline That Survived 40x Traffic Spikes on Event Nights

Based on a delivery engagement with StageNorth Live

40x

TRAFFIC SPIKE ABSORBED

-52%

INFRASTRUCTURE COST

6 to 0

EVENT-NIGHT INCIDENTS

<1 s

API LATENCY AT PEAK

A fillable practice edition: the full story, 8 actions,
6 reflections and a 90-day plan - work it with your team.

Free - no email required
emersontechnology.com

MAKE THIS PRACTICE BOOK YOURS

Work it, do not just read it

This practice book turns the StageNorth Live case study into a working session for your team. Read the full story first - what happened, what was built and what it changed. Then tick the 8 actions you can commit to, write short answers to the 6 reflections, and finish with a 90-day plan. Every field in this PDF is fillable: open it in Adobe Acrobat Reader, Chrome, Edge or macOS Preview, type directly into the boxes and save.

YOUR NAME	ROLE
COMPANY	TEAM / DEPARTMENT
DATE	EMAIL (OPTIONAL)

MY GOAL FOR THIS PRACTICE SESSION

- 01 The story** - What happened at StageNorth Live - before, during and after the project.
- 02 The results** - The numbers that changed, plus industry benchmarks from public research.
- 03 The 8-action checklist** - Tick boxes as your organization completes each step. Owner and date fields included.
- 04 The 6 reflections** - Fillable questions that pressure-test your own roadmap.
- 05 My 90-day plan** - Three fillable phase cards that turn this case study into your roadmap.

RUN IT AS A 90-MINUTE TEAM SESSION

25 min READ Read the story together. Ask everyone to note the one number that surprised them most.	40 min PRACTICE Work the 8-action checklist and the 6 reflections. Assign an owner and a date to every action you commit to.	25 min COMMIT Fill in the 90-day plan. One goal per phase, one number that tells you it worked. Book a review date before you leave the room.
--	--	---

THE STORY

What happened, and why it worked

01 The Company

StageNorth Live produces streamed events and ticketing for independent venues: concerts, comedy shows, festival weekends and the on-sale moments that sell them out. The company handles the full commercial loop for its venue partners — event pages, ticket inventory, checkout, live-stream delivery and the interactive layers such as polls and leaderboards that keep remote audiences engaged during shows. With a 70-person team, StageNorth operates in a market where the difference between a smooth on-sale and a crashed one is measured in sellouts versus apologies, and where venue partners switch platforms quickly if their biggest nights fail. The company's traffic profile is unlike ordinary e-commerce, and that difference defines everything about its infrastructure problem. Demand arrives in violent bursts: a popular on-sale can multiply traffic 40 times above baseline within minutes, live events concentrate thousands of simultaneous viewers into poll votes and leaderboard refreshes, and then demand collapses back to near-nothing until the next event. A platform built for the average day is structurally wrong for that profile — which is exactly the platform StageNorth had.

02 Where Things Stood

StageNorth's website collapsed every time a popular on-sale went live. The infrastructure was a fixed fleet of virtual machines sized for average traffic, so the 40x spikes around event nights — flash on-sales, live polls, real-time leaderboards — overwhelmed databases and API servers within minutes. The failure sequence was familiar to everyone in the company: the on-sale opened, traffic surged, the API servers queue built up, the database exhausted its connections, and checkout failed for the very fans who most wanted tickets. One crashed on-sale for a sold-out comedy festival cost an estimated \$750,000 in lost ticket revenue and generated a wave of social-media complaints that venue partners read the next morning.

The human cost was chronic as well as acute. Engineers spent event nights manually restarting services instead of watching the show, following restart scripts that sometimes worked and the operating team treated event-night survival as a staffing problem rather than an architecture problem. Capacity planning offered no good options: adding capacity for peaks meant paying for idle servers the other 25 days a month, while sizing for economics meant accepting that the biggest commercial moments would be the biggest failures. Six event-night incidents stood in the record, each one damaging a relationship with a venue partner whose own reputation was on the line that night. Gartner has reportedly predicted that by 2025 more than 95% of new digital workloads will be deployed on cloud-native platforms, up from 30% in 2021 — and StageNorth's fleet of fixed VMs was a textbook example of the workload pattern that prediction says is disappearing.

03 The Challenge

The challenge was architectural geometry: the workload's peak-to-average ratio made every conventional option wrong. Sizing for the peak wasted money for most of the month; sizing for the average guaranteed failure at the moments that mattered most. AWS engineering reports cite Prime Day events where DynamoDB peaked at tens of millions of requests per second — on-demand scaling absorbing bursts without pre-provisioning — and that pattern describes the capability StageNorth needed: capacity that appears during the burst and disappears after it, billed per use rather than per hour of idle fear.

Beyond scaling, the failure history exposed deeper design flaws that raw capacity could not fix. Checkout was a multi-step flow — seat hold, payment, confirmation — with no durable state between steps, so a mid-flow crash stranded held seats and produced the manual-restart ritual that consumed event nights. Writes were not idempotent, so a network retry could double-charge a fan or double-count a vote, the kind of defect that surfaces

publicly at the worst possible moment. And the monitoring watched servers rather than business outcomes, so dashboards stayed green while conversion collapsed. Datadog's State of Serverless research has reportedly found a majority of AWS-using organizations run Lambda in production, with adoption rising year over year — which suggested the tooling for the fix was mature. The challenge was to re-architect the event pipeline so that bursts became routine, failures became retries, and event nights became boring.

04 Discovery and Diagnosis

Emerson Technology began with a traffic-profile analysis across a full event cycle rather than an average day, charting hourly load for a representative month that included two on-sales and one streamed show. The data quantified what the team knew anecdotally: baseline traffic occupied a narrow band, while event-night traffic spiked 40x within minutes of an on-sale opening and decayed over hours. The profile also showed where the old fleet failed first — API servers saturating before the database, the database exhausting connections before the checkout flow completed, and static asset requests competing with transactional API calls for the same bandwidth.

The code and workflow diagnostic mapped the failure modes that scaling alone would not cure. The checkout saga had no durable state machine: if the payment step failed after a seat was held, recovery depended on which process was alive and which cron job had last run, which is why engineers carried restart scripts. Order events flowed through tightly coupled calls, so a slow downstream service backed pressure up into the request path. Write paths lacked idempotency keys, leaving retries ambiguous. The diagnosis ended with a target architecture shaped by the traffic profile itself: a pay-per-use request path that scales to zero between events, edge caching to absorb repetitive viewing traffic before it reaches compute, durable orchestration for multi-step flows, and a load-testing regime that treats every major on-sale as a rehearsed launch rather than a live experiment.

05 The Solution

We rebuilt the event pipeline on AWS serverless services, designing every layer around the same principle: capacity follows demand, state survives failure, and the edge absorbs what the compute never needs to see.

The request path

API Gateway and Lambda serve ticket search and checkout APIs, scaling per request with no fleet to pre-warm. DynamoDB runs in on-demand capacity mode for inventory and seat holds, absorbing burst traffic without capacity-planning calls and scaling back to zero between events. CloudFront caches static assets and price-sorted listings at the edge, so the vast majority of event-night traffic — repetitive viewing of the same pages and listings — never reaches Lambda at all. Most event-night traffic is repetitive viewing, not unique work, and caching it at the edge is what keeps the transactional path fast for the requests that genuinely need compute. Payment processing stayed on Stripe, with webhooks fanned out through EventBridge so consuming systems subscribe to the events they care about.

The checkout saga

Step Functions orchestrates the checkout saga — seat hold, payment, confirmation — as an explicit state machine with idempotent retries replacing the manual restart scripts of the old stack. Every write carries an idempotency key, so a network retry can no longer double-charge a fan or double-count a vote; the workflow simply resumes from its last durable state. A failed payment releases the held seat automatically after a defined window instead of stranding it in limbo, and EventBridge plus SQS carry order events to downstream consumers so a slow service queues rather than cascades. Queues turn crashes into ordinary retries, and state machines turn multi-step flows into inspectable, testable workflows.

The rehearsal culture

Infrastructure is defined in Terraform with per-event scaling profiles, so a marquee on-sale and a Tuesday show get appropriately different alert thresholds and quota headrooms. A load-testing suite simulates on-sale bursts before every major event — login floods, virtual queues, payment retries — because rehearsals consistently surface downstream limits that architecture diagrams never show, and each finding is fixed cheaply before the real night. Monitoring was re-pointed at business metrics: checkout conversion, queue depth and error rates during live events, rather than raw server health, which often stays green right up until customers start complaining publicly. And spend is bounded by design — alarms and service quotas cap the scaling budget per event, so a runaway loop or bot surge pages an engineer before it pages the invoice.

06 The First 90 Days

The first month rebuilt the foundations: the request path moved to API Gateway and Lambda, DynamoDB on-demand tables replaced the overloaded database for inventory and seat holds, and CloudFront was deployed in front of everything cacheable. The first event night on the new path was deliberately a mid-size on-sale, not the biggest show of the season — a rehearsal in production, with the legacy stack still available and a rollback plan on the shelf. It ran clean, and the team used the following two weeks to migrate the checkout flow onto Step Functions with idempotency keys on every write.

Weeks five through eight delivered the interactive layer: AppSync WebSockets began pushing live poll results to thousands of viewers at once, replacing the polling loop that used to hammer the API during shows. The load-testing suite came online in the same period, and its first full rehearsal of a marquee on-sale burst surfaced a connection limit in a downstream service that no design review had flagged — the exact class of discovery rehearsals exist for. Weeks nine through twelve hardened operations: per-event scaling profiles were configured in Terraform, business-metric alerting replaced server dashboards as the event-night source of truth, and the scaling-budget alarms were tuned with finance so spend caps matched revenue expectations for each show. By day 90, the team had run three on-sales on the new platform, the restart scripts had been retired to documentation history, and engineers had started attending the events they shipped — as spectators.

07 The Results

The next four major on-sales ran without a single outage, absorbing 40x baseline traffic with sub-second API responses — including a rescheduled date for the same sold-out comedy festival whose crashed on-sale had once cost an estimated \$750,000 in projected revenue. That on-sale now clears its inventory in minutes. Infrastructure costs fell 52% because the platform no longer pays for idle peak capacity; it scales to zero between events and pays per request when the burst arrives. Event-night engineering shifts disappeared entirely, and the team ships new interactive features, like live polls, in days rather than the multi-week capacity projects the old fleet required.

METRIC	BEFORE	AFTER
Traffic spike absorbed	-	40x
Infrastructure cost	Baseline	-52%
Event-night incidents	6	0
API latency at peak	-	Under 1 s

The qualitative changes show up in the company's calendar and its culture. Event nights moved from the engineering on-call rotation to the marketing calendar — launches are now commercial moments rather than technical emergencies. Venue partners stopped asking nervously about the platform's reliability, because two

clean seasons answered the question more persuasively than any sales deck. Feature development accelerated in a way the old architecture structurally forbade: new interactive experiences ship behind feature flags and scale automatically with audience size, so the constraint on product ambition is no longer what the fleet can survive. The platform finally matches the shape of the business it serves — mostly quiet, occasionally enormous, never fragile.

08 What Your Business Should Do

StageNorth's rebuild is the template for any bursty workload, and it starts with honesty about your traffic. Profile traffic before choosing architecture: chart hourly load across a full event cycle, not an average day, because spiky workloads justify managed, pay-per-use scaling far more cheaply than any pre-provisioned server fleet. Cache aggressively at the edge — push static assets, listings and read-heavy endpoints to a CDN so bursts are absorbed before reaching your compute layer, since most burst traffic is repetitive viewing, not unique work. Make every write idempotent, designing checkouts, votes and holds so retries cannot double-charge or double-count; idempotency keys turn network flakiness into a routine non-event instead of an overnight support queue. Choose on-demand databases for bursts, using capacity modes that scale to zero and absorb peaks without capacity-planning calls — paying per request beats paying for idle servers 25 days a month.

Then rehearse until the big night is ordinary. Load-test like it is on-sale night: simulate real burst patterns such as login floods, virtual queues and payment retries before every major event, because rehearsals consistently surface downstream limits that architecture diagrams never show. Decouple steps with durable workflows, orchestrating multi-step flows like hold, pay and confirm with state machines so a failed payment never loses a held seat — queues turn crashes into ordinary retries. Alert on business metrics rather than servers: watch checkout conversion, queue depth and error rates during live events, because infrastructure dashboards often stay green right up until customers start complaining publicly. And set a scaling budget per event, capping spend with alarms and service quotas, since serverless bills per use and a runaway loop or bot surge should page someone before the invoice does. As you plan, interrogate your own traffic honestly: what your last big spike actually cost in revenue, refunds and reputation; how much idle capacity you pay for monthly to protect against peaks lasting hours; where a payment retry would create a duplicate order right now; and which of your systems would fail first under 40x load — and whether you have ever tested it. Work through these eight steps with your team using the action checklist in this practice book, and revisit the reflection questions each quarter so the platform keeps scaling with the ambition of the business rather than the size of its fears.

REAL-WORLD PROOF & BENCHMARKS

Gartner has reportedly predicted that by 2025 more than 95% of new digital workloads will be deployed on cloud-native platforms, up from 30% in 2021. - Gartner

AWS engineering reports cite Prime Day events where DynamoDB peaked at tens of millions of requests per second — on-demand scaling absorbing bursts without pre-provisioning. - AWS, reported 2023

Datadog's State of Serverless research has reportedly found a majority of AWS-using organizations run Lambda in production, with adoption rising year over year. - Datadog State of Serverless

THE PRACTICE PAGES

The 8-action checklist

Work through the list with your team. Tick each box as your organization completes the step, and fill in an owner and a target date so every action has a name and a deadline.

☐

1. Profile traffic before choosing architecture

Chart hourly load across a full event cycle, not an average day. Spiky workloads justify managed, pay-per-use scaling far more cheaply than any pre-provisioned server fleet.

OWNER

TARGET DATE

STATUS

--	--	--

☐

2. Cache aggressively at the edge

Push static assets, listings and read-heavy endpoints to a CDN so bursts are absorbed before reaching your compute layer. Most event-night traffic is repetitive viewing, not unique work.

OWNER

TARGET DATE

STATUS

--	--	--

☐

3. Make every write idempotent

Design checkouts, votes and holds so retries cannot double-charge or double-count. Idempotency keys turn network flakiness into a routine non-event instead of an overnight support queue.

OWNER

TARGET DATE

STATUS

--	--	--

☐

4. Choose on-demand databases for bursts

Use capacity modes that scale to zero and absorb peaks without capacity-planning calls. Paying per request beats paying for idle servers 25 days a month.

OWNER

TARGET DATE

STATUS

--	--	--

☐

5. Load-test like it's on-sale night

Simulate real burst patterns such as login floods, virtual queues and payment retries before every major event. Rehearsals consistently surface downstream limits that architecture diagrams never show.

OWNER

TARGET DATE

STATUS

--	--	--

**6. Decouple steps with durable workflows**

Orchestrate multi-step flows like hold, pay and confirm with state machines, so a failed payment never loses a held seat. Queues turn crashes into ordinary retries.

OWNER

TARGET DATE

STATUS

--	--	--

**7. Alert on business metrics, not servers**

Watch checkout conversion, queue depth and error rates during live events rather than raw server health. Infrastructure dashboards often stay green right up until customers start complaining publicly.

OWNER

TARGET DATE

STATUS

--	--	--

**8. Set a scaling budget per event**

Cap spend per event with alarms and service quotas. Serverless bills per use, so a runaway loop or bot surge should page someone before the invoice does.

OWNER

TARGET DATE

STATUS

--	--	--

AFTER THE CHECKLIST: OUR TOP 3 PRIORITIES

From the eight actions above, write down the three you will start first - and the single number you expect each one to move. Three priorities with owners beat eight without any.

PRIORITY 1	PRIORITY 2	PRIORITY 3
NUMBER IT SHOULD MOVE	NUMBER IT SHOULD MOVE	NUMBER IT SHOULD MOVE

THE PRACTICE PAGES

Reflections for your team

Use these questions in your next planning session and type short answers into the boxes. The gaps you find are usually the fastest wins.

Q1. What did your last big traffic spike actually cost you — in revenue, refunds and reputation?

Q2. How much idle capacity do you pay for every month to protect against peaks that last hours?

Q3. Where in your flow would a payment retry right now create a duplicate order?

Q4. Who is on call during your next launch, and what can they actually fix at 2 a.m.?

Q5. Which of your systems would fail first under 40x load — and have you ever tested it?

Q6. Could your engineers ship a new interactive feature during event week, or does every change freeze?

THE PRACTICE PAGES

My 90-day plan

Finish the session by committing to a plan. Three phases, one goal each, one number that tells you it worked. Bring this page to your next leadership review - or to a free consultation with Emerson Technology, and we will pressure-test it with you.

Days 1-30 Focus: audit and alignment

PRIMARY GOAL

KEY ACTIONS

HOW WE MEASURE SUCCESS

OWNER

Days 31-60 Focus: build and prove

PRIMARY GOAL

KEY ACTIONS

HOW WE MEASURE SUCCESS

OWNER

Days 61-90 Focus: scale and measure

PRIMARY GOAL

KEY ACTIONS

HOW WE MEASURE SUCCESS

OWNER

Tip: save a copy per team so each group keeps its own answers. Revisit the reflections every quarter - the questions stay the same even as the numbers move.

BOOK THE REVIEW BEFORE YOU CLOSE THE BOOK

Plans without a review date drift. Before you close this practice book, agree on when you will come back to these pages, who will run that session, and when you will check the numbers again. If you want an outside pair of eyes, bring this page to a free consultation with Emerson Technology and we will pressure-test the plan with you.

REVIEW DATE	REVIEWED BY	NEXT CHECK-IN
------------------------------------	------------------------------------	--------------------------------------

Ready for a transformation like StageNorth Live's?

Emerson Technology designs and builds the data, AI, cloud and automation systems behind results like these. On a free consultation call we will map this exact playbook to your business - your data, your constraints, your numbers.

Book a free consultation

support@emersontechnology.com

or start online: emersontechnology.com/contact