Scaling up Psycholinguistics with Pushkin

Cherrie Chang[1] & Joshua Hartshorne[1]
Communication Sciences & Disorders, MGH IHP

This is Alexander Pushkin, a really famous and revolutionary Russian literarist

This is Pushkin, a platform for running online experiments that we hope will be as revolutionary (and famous, maybe) as Alexander Pushkin was

Introduction

Goal:

Develop a free, open source platform– **Pushkin** – that fully integrates building, hosting and managing multiple **massive online experiments (MOEs)** in one site.

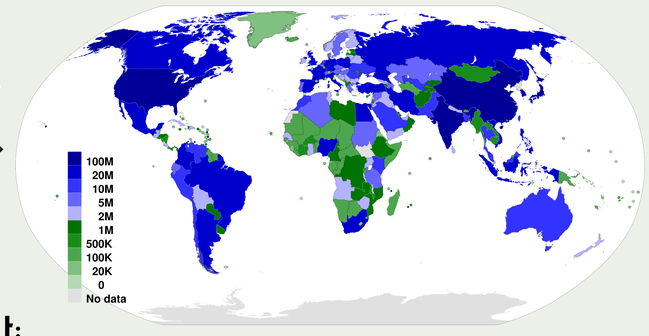
Why Massive Online Experiments (MOEs):

#1: Sampling scale is a significant bottleneck of psycholinguistics research.

- A good (read: replicable) study needs a large sample that satisfies (Hartshorne, 2019): 1) stimulus set comprehensiveness, 2) demographic diversity, 3) statistical power, etc.
- This is further multiplied by individual differences and developmental questions.
- But it quickly becomes expensive and hard to recruit large samples in the lab or online.

👉 **MOEs (in theory):**

- Sample internet-scale populations
- i.e. over 70% of the world's population (~6 billion) →

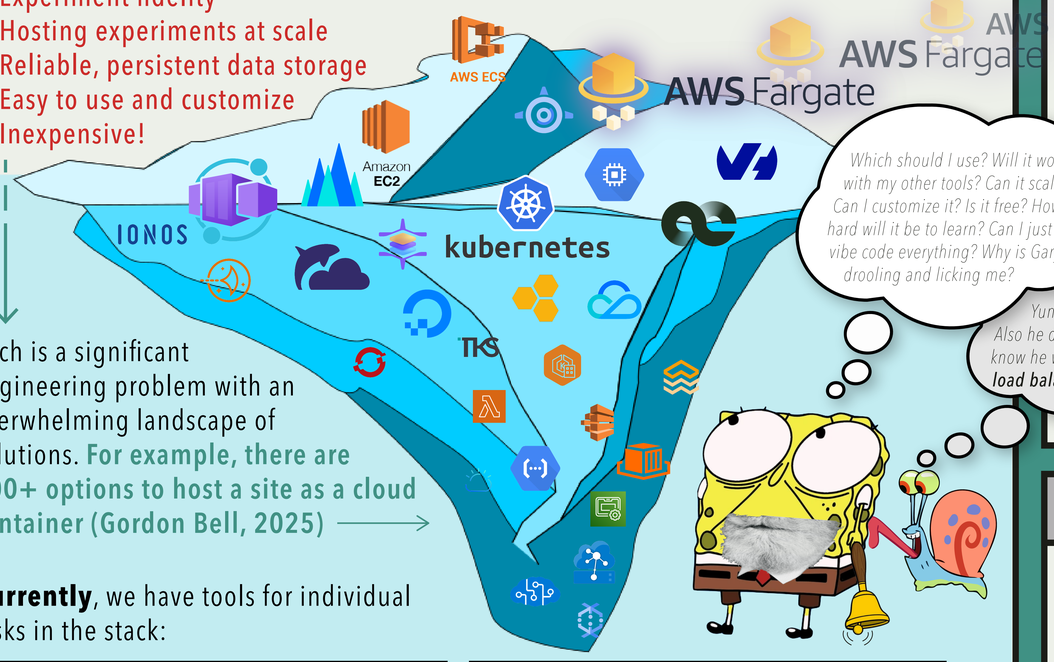


This seems obvious.
But in practice...

Why Nobody Has Done It:

#2: Massive online experiments present a significant, difficult engineering problem, and current tools are ill-equipped to handle it.

- A good platform for supporting MOEs faces many challenges:
  1. Experiment fidelity
  2. Hosting experiments at scale
  3. Reliable, persistent data storage
  4. Easy to use and customize
  5. Inexpensive!



- Each is a significant engineering problem with an overwhelming landscape of solutions. **For example, there are 100+ options to host a site as a cloud container (Gordon Bell, 2025)** →
- **Currently**, we have tools for individual tasks in the stack:

- But none of these tools can satisfy all the needs of running MOEs.
  - E.g. Qualtrics mainly does survey-style studies and struggles with precise timing.
  - E.g. Gorilla is pay by subject, which becomes very expensive at internet-scale.
  - E.g. PCIbex would struggle to meet internet-scale traffic loads.
- It is also still daunting to figure out which tool to choose out of many for each task, especially in terms of whether they will fit together.
- Researchers may not know all the tech involved, e.g. a load balancer to balance traffic.

So we need a full stack platform that tightly integrates:
experiment building + deployment & hosting + data storage
While still being easy to use, customizable and inexpensive.

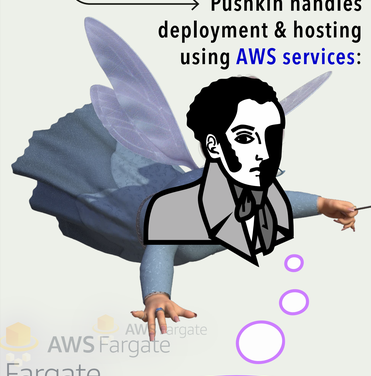
Design

Researcher flow:

Reads the Pushkin docs, uses the Pushkin CLI to

pushkin-cli
1. Create site and experiments
2. Prep experiments as Docker images
3. Run database migrations to set up database schemas and seed stimuli

→ **Pushkin handles deployment & hosting using AWS services:**



Don't worry, mother knows best!
- *Fargate auto-scales containers to handle traffic spikes, and is pay-per-use*
- *CloudFront serves worldwide and fast*
- *RDS manages databases with automatic backups*
- ***All you need to do to set this up is run `pushkin-aws init`***

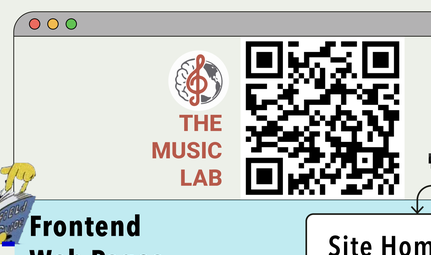
Route53 (DNS)
|
Cloudfront (+WAF firewall)
|
S3 Bucket (serves static website files)

AWS Fargate (Serverless containers)

RDS (Databases)

Participant flow:

Sees cool Pushkin site like Sam Mehr's The Music Lab, clicks



Frontend Web Pages (React + Redux)
- Site Homepage
- navigate to
- Experiments Page
- **Exp 1** (jsPsych) **Server**
- **Exp 2** (jsPsych) **Server**
- **Exp 3** (jsPsych) **Server**

Load Balancer (AWS ALB)

Send HTTP requests
1. startExperiment
2. getStimuli
3. getResponse
4. getModelPrediction (conditional stimulus assignment based on response)
5. saveResults

Backend (React + Redux)
- **API Server** (Express/Node.js)
- Forward HTTP requests (AMQP)
- **Message Queue** (RabbitMQ)
- (One worker for each experiment)
- **Exp 1 Worker** (Node + Python)
- **Exp 2 Worker** (Node + Python)
- **Exp 3 Worker** (Node + Python)
- Query and write to databases (Knex)

Main Database
Tables shared across experiments:
users + userResults + userMeta
Tables per exp: stimuli + responses
(linked by pseudonymous UUIDs)

Transactions Log
Records every database query for auditing

Databases (PostgreSQL)

Updates

#1: Longitudinal Studies!

- Persistent login profiles via Auth0: participants return across devices/sessions with the same user ID.
  - Pseudonymous: experiment tables only store UUIDs; PII (email/name) isolated to one linkable table
  - Dual-mode: can choose anonymous (cookie-based) or authenticated for longitudinal designs

#2: Codebase Modernization

- Migrated to monorepo using modern AWS SDK code.
- Refactored into testable modules with unit tests.

#3: Load Testing

- Compared baseline config performance to "one tier up".
  - Baseline: 100% success up to ~10 concurrent users
  - One Tier Up: better latency (P95 66–176ms), sustained to ~15 users, graceful degradation after
  - Both handled 300 simultaneous logins at 100% success.

Next Steps:

- Horizontal scaling to enable proper viral load
- Get critical mass and encourage contributions

Funding: We are thankful to be funded by the NIH on the Pathways to Enable Open Science Ecosystems grant (POSE-II; ID: GR1000122).

References

- Gordon Bell, A. (2025, March 24). 105 Ways to Run Containers: The Cloud Container Iceberg. Pulumi Blogs.
- Hartshorne, J. K., de Leeuw, J. R., Goodman, N. D., Jennings, M., & O'Donnell, T. J. (2019). A thousand studies for the price of one: Accelerating psychological science with Pushkin. Behavior Research Methods, 51(4),1782–1803. DOI: 10.3758/s13428-018-1155-z

Digital version of poster: