QCon NY 2016 Recap

Matt Dickenson
Ibotta
June 30, 2016

Themes

- 🏡 Architecture & Services
- 👭 Distributed Systems
- ⚽ Culture & Teams
- 🤖 Machine Learning
- 🗣️ Hallway Track

Server-Side Scripting

Katharina Probst (@probst_kathrin), Netflix

- Netflix runs JS clients on 1,000s of device types
- Devices sometimes require API customization
    - Different interaction patterns (remote, voice, swipe)
- Device teams uploaded Groovy scripts to the server
- Until one day a memory-intensive script broke stuff

Server-Side Scripting

Katharina Probst (@probst_kathrin), Netflix

- “Does the *current system* meet the *current requirements*?”
- Flexibility: A/B testing specific to devices, for example
- Velocity: decoupling deployments of multiple teams
- Resiliency: this was a weak point of uploading scripts
- Developer experience: switched to containers

Server-Side Scripting

Katharina Probst (@probst_kathrin), Netflix

- When language shifted from “script” to “app” it signaled a change in usage patterns

- When allowing others to execute code in your environment, ensure that responsibility correlates with freedom

- Don’t be afraid to change your system as requirements change

Thousands of Services

Matt Ranney (@mranney), Uber

- Costs of Migrating to Service-Oriented Architecture
- Complexity of a distributed system
- Tempting to build around problems rather than fix them
- Polyglot dev team fragments culture, making it hard to share code and move between teams

Thousands of Services

Matt Ranney (@mranney), Uber

- Performance: easy to believe an operation is fast because *your part* is fast
- Fan out
    - If something takes 1ms on average but 1s in the 99th percentile, and you use it 100 times, there's a 63% chance you'll hit the worst case
- Context: often gets propagated across service calls because no one knows what they can safely remove

Thousands of Services

Matt Ranney (@mranney), Uber

- Failure testing: adding this retroactively is challenging both technically and culturally

- Migrations: use a “carrot” and not a “stick”

- Politics: “developers are willing to increase complexity in exchange for not having hard conversations"

Mesos & Containers

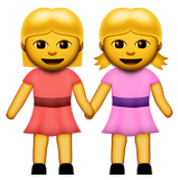
David Greenfield, Two Sigma

- Hedge funds do lots of simulations (data pipeline)
- FIFO work scheduling ignores human idiosyncrasies
- Rolling out an internal system
    1. Have a couple users & let them talk to teach other
    2. Achieve previous SLA (no improvements!)
    3. Do the easiest possible thing to get “wow” factor
    4. Do the cool things you wanted to do in the first place

Mesos & Containers

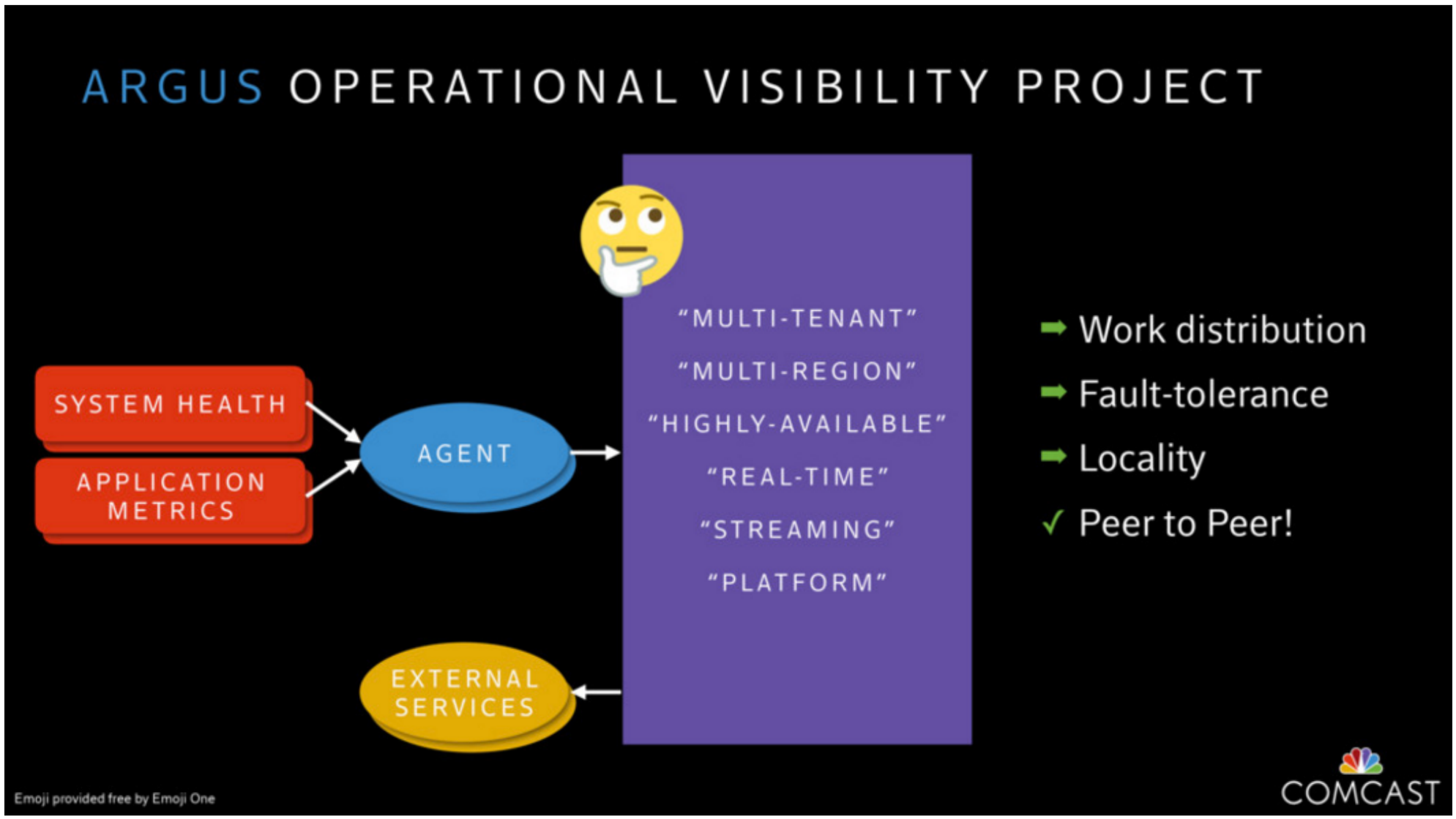
David Greenfield, Two Sigma

- Hedge funds do lots of simulations (data pipeline)
- FIFO work scheduling ignores human idiosyncrasies
- Rolling out an internal system
    1. Have a couple users & let them talk to teach other
    2. Achieve previous SLA (no improvements!)
    3. Do the easiest possible thing to get “wow” factor
    4. Do the cool things you wanted to do in the first place

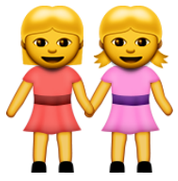


Membership Protocols

Sean Cribbs (@seancribbs), Comcast



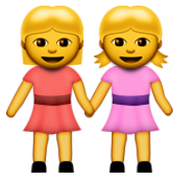
Peer-to-peer: majority of participants share the same role



Membership Protocols

Sean Cribbs (@seancribbs), Comcast

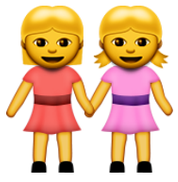
- Desirable properties of a membership protocol
  - Connectedness (not necessarily direct)
  - Balance (no single node can 🔥 the whole cluster)
  - Short path length
  - Low clustering
    - If A sends to B & C, don’t want B to also send to C
  - Accuracy: detect failed nodes quickly



Membership Protocols

Sean Cribbs (@seancribbs), Comcast

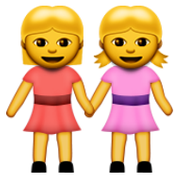
- Several popular options available
- SWIM
  - Round-robin approach to failure detection (❤️beat)
  - If A doesn’t get an ack from B, A asks C and D if they think B is alive



Membership Protocols

Sean Cribbs (@seancribbs), Comcast

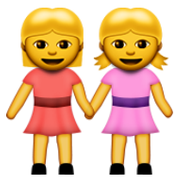
- SCAMP
- Partial-view, self-organizing cluster
- Nodes join via a two-stage random walk
- Random walk to another node to use as starting point
- Join message propagates to random # of nodes



Membership Protocols

Sean Cribbs (@seancribbs), Comcast

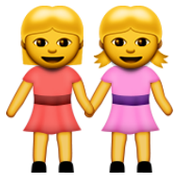
- CYCLON
  - On a regular interval, shuffle your view with your oldest neighbor
  - Helps maintain balance
  - Sort of like certain versions of poker



Membership Protocols

Sean Cribbs (@seancribbs), Comcast

- Dissemination Protocols: “Epidemic Broadcast Trees”
  - All nodes start with an eager push
  - Broadcast triggers eager push
  - Duplicate messages result in pruning: if A and B send the same message to C and A gets there first, B moves into its "lazy" set
  - Over time this results in a spanning tree
  - Lazy-push sends "I have a message" notifications. If B's "I have" message reaches C before A's eager push, a link between C and B is grafted back into the tree (lazy pushes can be batched)



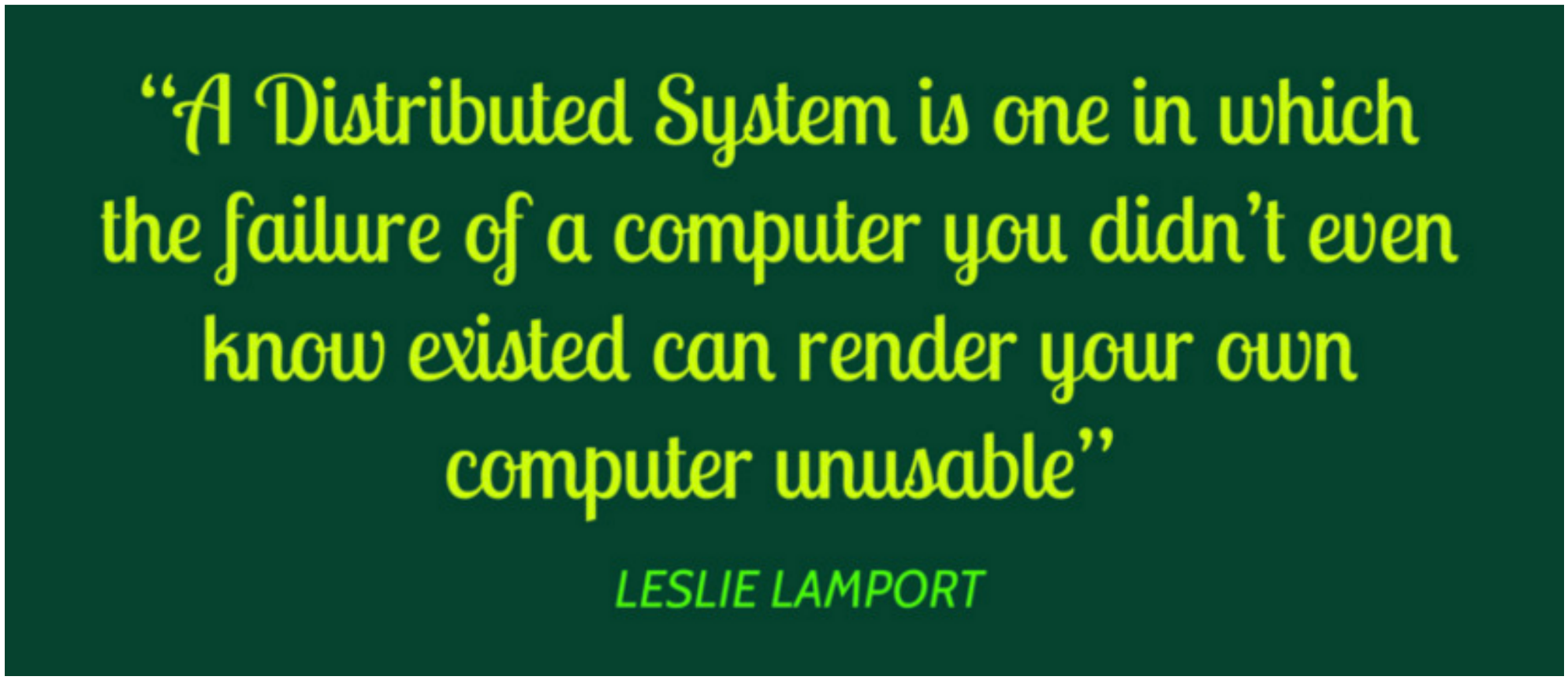
Membership Protocols

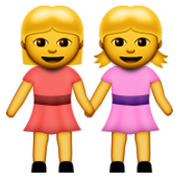
Sean Cribbs (@seancribbs), Comcast

- Most systems don’t need weakly consistent membership (can use DNS)
- Between research paper and implementation there is a “gulf of despair”
- Cassandra uses Hybrid Partial View
- Riak uses PLuM trees

Verification of DS

Caitie McCaffrey (@caitie), Twitter





Verification of DS

Caitie McCaffrey (@caitie), Twitter

Mark Imbriaco @markimbriaco · 8 Dec 2014

You know what's worse than computers? Computers talking to other computers.

20 21

DHH

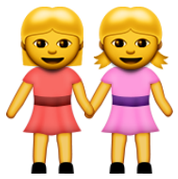
@dhh

Follow

@markimbriaco First law of distributed computing: Don't distribute your computing!

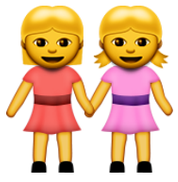
RETWEETS 2

LIKES 5



Verification of DS

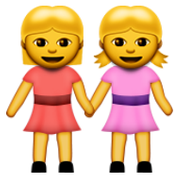
Caitie McCaffrey (@caitie), Twitter



Verification of DS

Caitie McCaffrey (@caitie), Twitter

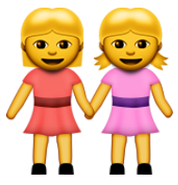
- Formal Verification
- TLA+
- Coq
- Amazon claims to have used formal verification to find bugs in S3, Dynamo, and EBS



Verification of DS

Caitie McCaffrey (@caitie), Twitter

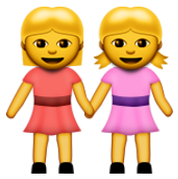
- Formal Verification
- TLA+
- Coq
- Amazon claims to have used formal verification to find bugs in S3, Dynamo, and EBS



Verification of DS

Caitie McCaffrey (@caitie), Twitter

- Unit Tests (still valuable in a distributed environment!)
- Types != Testing
  - “TCP doesn’t care about your type system”
- Integration tests (simple 3-node setup)
- Property-based testing (QuickCheck)



Verification of DS

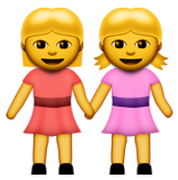
Caitie McCaffrey (@caitie), Twitter

- Fault injection (Netflix)
  - Chaos Monkey
    - kills instances
  - Latency Monkey
    - artificial delays
  - Chaos Gorilla
    - AZ outage

Verification of DS

Caitie McCaffrey (@caitie), Twitter

- Game Days
  - Amazon
  - Stripe

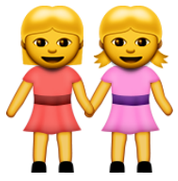


Verification of DS

Caitie McCaffrey (@caitie), Twitter

- Game Days

1. Notify team
2. Induce failures
3. Monitor systems
4. Observing team monitors recovery, files bugs
5. Prioritize bugs & fix



Verification of DS

Caitie McCaffrey (@caitie), Twitter

- Verify critical components any way you can!
    - Formal verification
    - Unit tests
    - Integration Tests
    - Property Testing
    - Fault injection

Culture-Focused Startup

Sunil Sadasivan (@sunils34), Buffer

- Culture == behaviors that are valued
- Can think of culture as your team's "default settings"
- Buffer's culture: buffer.com/transparency
    - remote work, positivity, self-improvement, clarity, reflection, gratitude, "live smarter not harder"

Culture-Focused Startup

Sunil Sadasivan (@sunils34), Buffer

- To hire for culture, your team has to know what it is
- “There are no balanced people, only balance teams”
- If you hire for culture fit, you should fire for it too

Culture-Focused Startup

Sunil Sadasivan (@sunils34), Buffer

- Building a diverse applicant pool: A/B testing job postings (“hacker” was found to be unfriendly)

- References:

  - *High Output Management* by Andy Grove

  - *Reinventing Organizations* by Frederic Laloux

Incident Response

John Allspaw (@allspaw), Etsy

- People, not tools, resolve incidents
- Lots of knowledge in people's heads about incident response
- Cognitive fixation: remembering a similar pattern
- Thematic vagabonding: rapid context switching

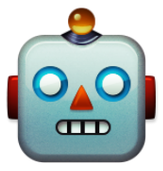
Incident Response

John Allspaw (@allspaw), Etsy

Incident Response

John Allspaw (@allspaw), Etsy

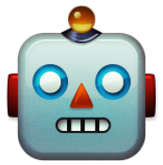
- Keep talking
- Bias toward peer review
- Read the error message
- Modeling exercise
- bit.ly/AllspawThesis



Building an AI Cloud

Simon Chan (@simonchannet), PredictionIO

- Simplicity-flexibility trade-off
- Approaches to simplification:
  - Automation
  - GUI
  - Code/scripts



Building an AI Cloud

Simon Chan (@simonchannet), PredictionIO

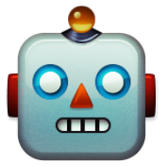
1. Define the goal (what constitutes a “good” prediction?)

2. Decide on the presentation (suggestive vs. decisive)

3. Import free-form data source (“life is more complicated than MovieLens”)

4. Construct features and labels (data transformation)

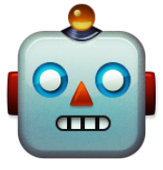
5. Set evaluation metrics (online & offline)



Building an AI Cloud

Simon Chan (@simonchannet), PredictionIO

6. Define “real-time” (daily? every 5 min?)
7. Find the right model (the fun part!)
8. Serve predictions (includes adding business logic)
9. Collect feedback for improvement (human learning)
10. Keep monitoring

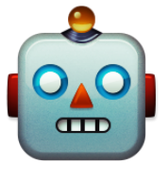


Vowpal Rabbit

John Langford, MS Research

- Explore-exploit trade-off
- Optimal stopping problem
- Multi-armed bandits

vs



Vowpal Rabbit

John Langford, MS Research

- Reinforcement learning
- Contextual bandits (regret minimization)
- Example at Skype: choosing which codex to use for a call
- Log as much as you can, because credit assignment becomes difficult

Hallway Track

- Lots of talk about distributed build systems
- “Can we wrap Mongo and Solr in a transaction?”
- “Most of the business logic is still in COBOL but we’re starting to add some Java.”

- *Everyone* is building distributed systems
- Can learn from systems with architectures different from ours
- Everything is a trade-off
- Easy to mistake technical decisions for cultural decisions and vice-versa