
Google™
11
I/O

Coding For The Cloud: How We Write Enterprise Apps for Google on App Engine

Ben Fried, Justin McWilliams, Eric Schoeffler, Justin Fagnani

May 11th, 2011

Hashtags: #io2011 #AppEngine #cloud #development

Session feedback: http://goo.gl/Td6HC

Google I/O 11

What we've written on App Engine

- OS and third-party software updates for one of the largest Enterprise Mac deployments in the world (Simian)
- 360 degree employee performance review and promotion for all of Google
- Help desk on-line knowledge base
- Course scheduling and management (CloudCourse)
- Corporate directory
- Financial and expense reporting tools
- Intranet home page
- Compensation review/display
- Desktop software management (AppReduce)
- Interview scheduling & feedback
- International payroll calculation
- Video conferencing unit fleet management

...and hundreds more

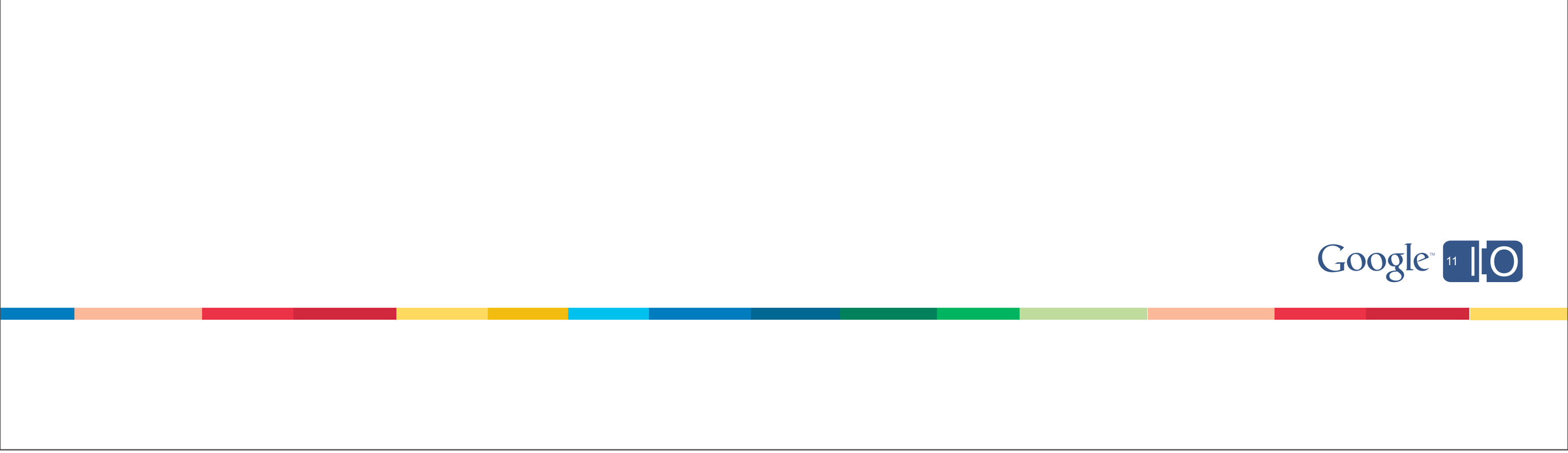
We Run Google on App Engine

Three Things for You to Take Away

Access from Anywhere

Scales Up, Scales Down, with Demand

Innovation Not Administration


Google
11

Justin McWilliams

Corporate Platforms Engineering

Pattern: App Engine as Web Service Platform

Not solely a user-facing website platform

- App Engine is HTTP server, but might not serve HTML
- Host web services for integration
- Consume other web services
- Various authentication options:
  - Google Account
  - OAuth
  - Custom, e.g. certificate based, shared secret, etc

Simian: OSX Software Deployment

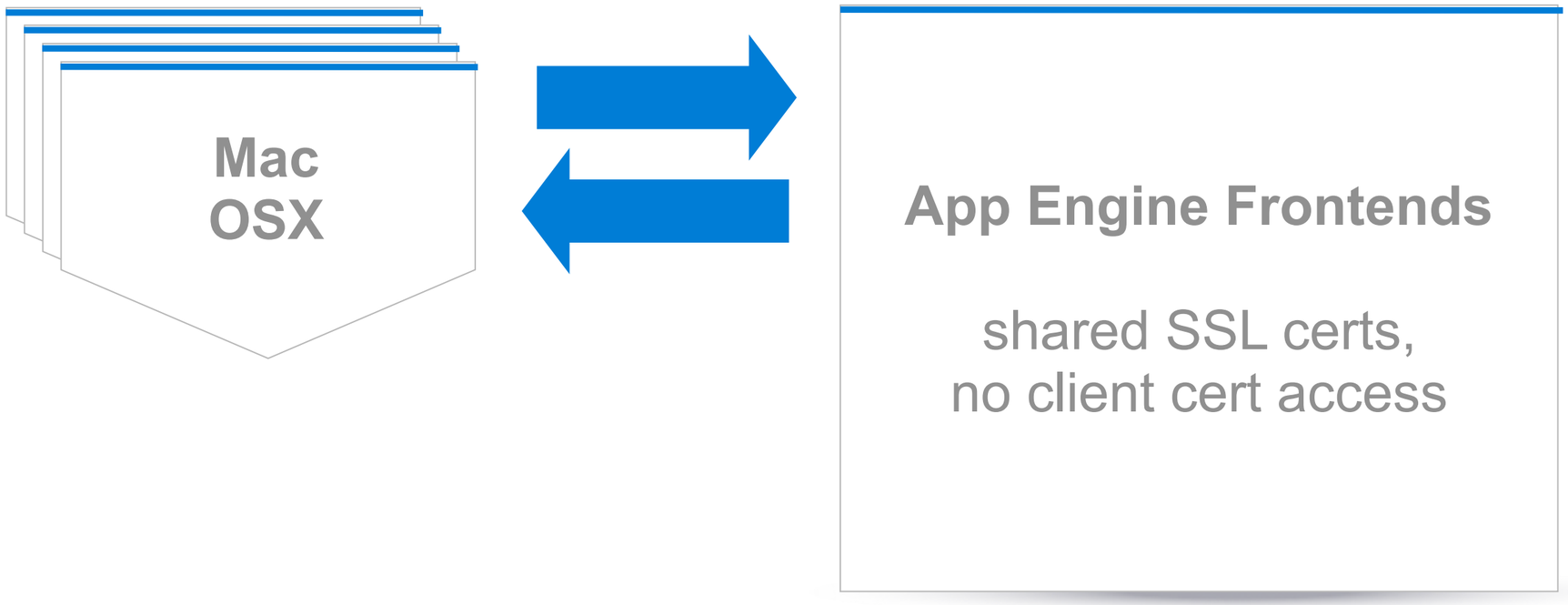
Simian was open-sourced at Macworld, January 2011: http://code.google.com/p/simian/

Simian: OSX Software Deployment

Simian was open-sourced at Macworld, January 2011: http://code.google.com/p/simian/

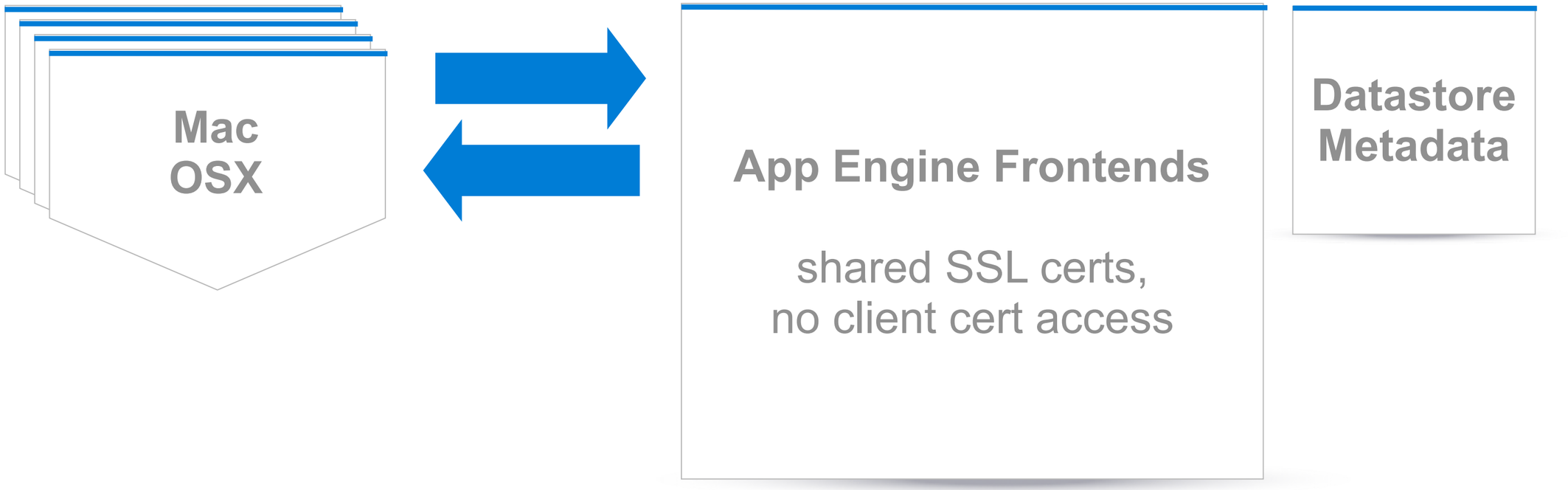
Simian: OSX Software Deployment

Simian was open-sourced at Macworld, January 2011: http://code.google.com/p/simian/



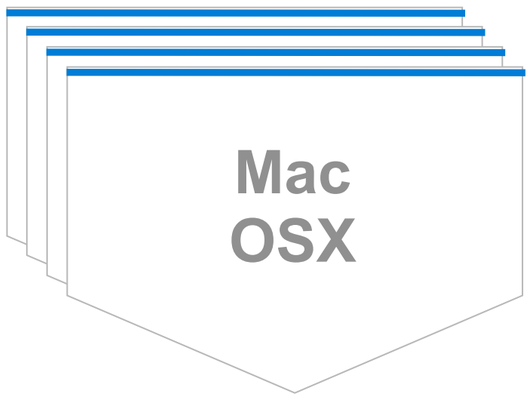
Simian: OSX Software Deployment

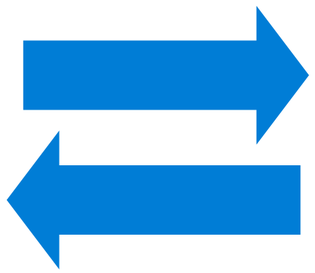
Simian was open-sourced at Macworld, January 2011: http://code.google.com/p/simian/

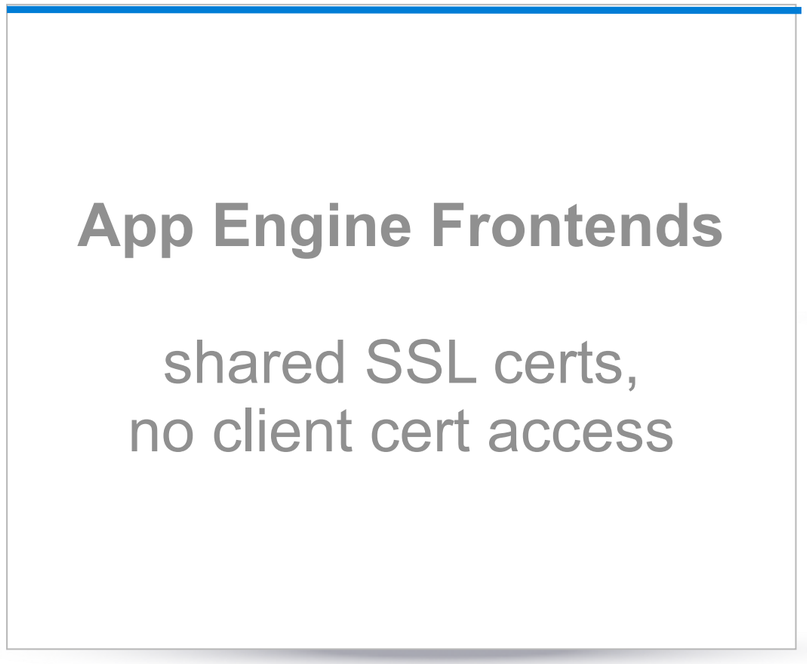


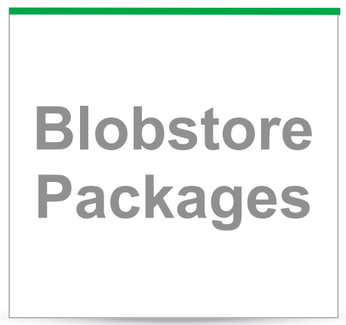
Simian: OSX Software Deployment

Simian was open-sourced at Macworld, January 2011: http://code.google.com/p/simian/



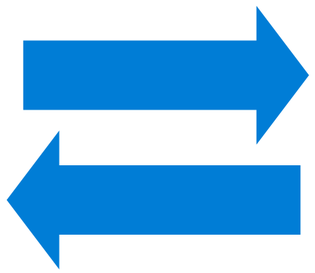






Simian: OSX Software Deployment

Simian was open-sourced at Macworld, January 2011: http://code.google.com/p/simian/



App Engine Frontends

shared SSL certs,
no client cert access

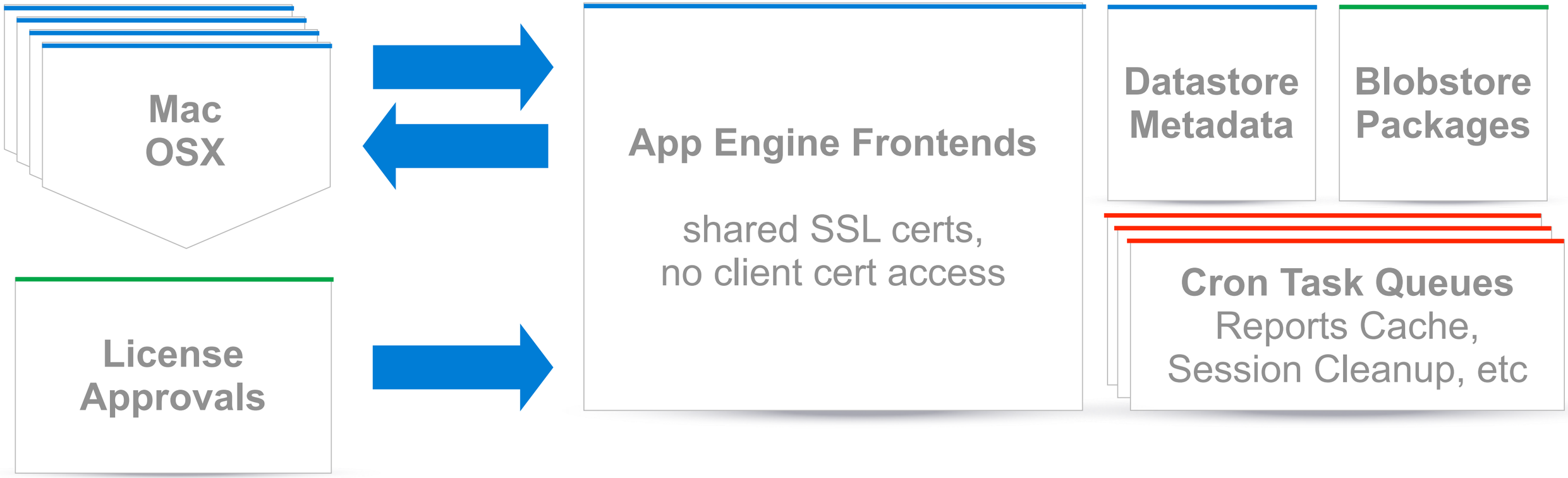
Datastore
Metadata

Blobstore
Packages

Cron Task Queues
Reports Cache,
Session Cleanup, etc

Simian: OSX Software Deployment

Simian was open-sourced at Macworld, January 2011: http://code.google.com/p/simian/



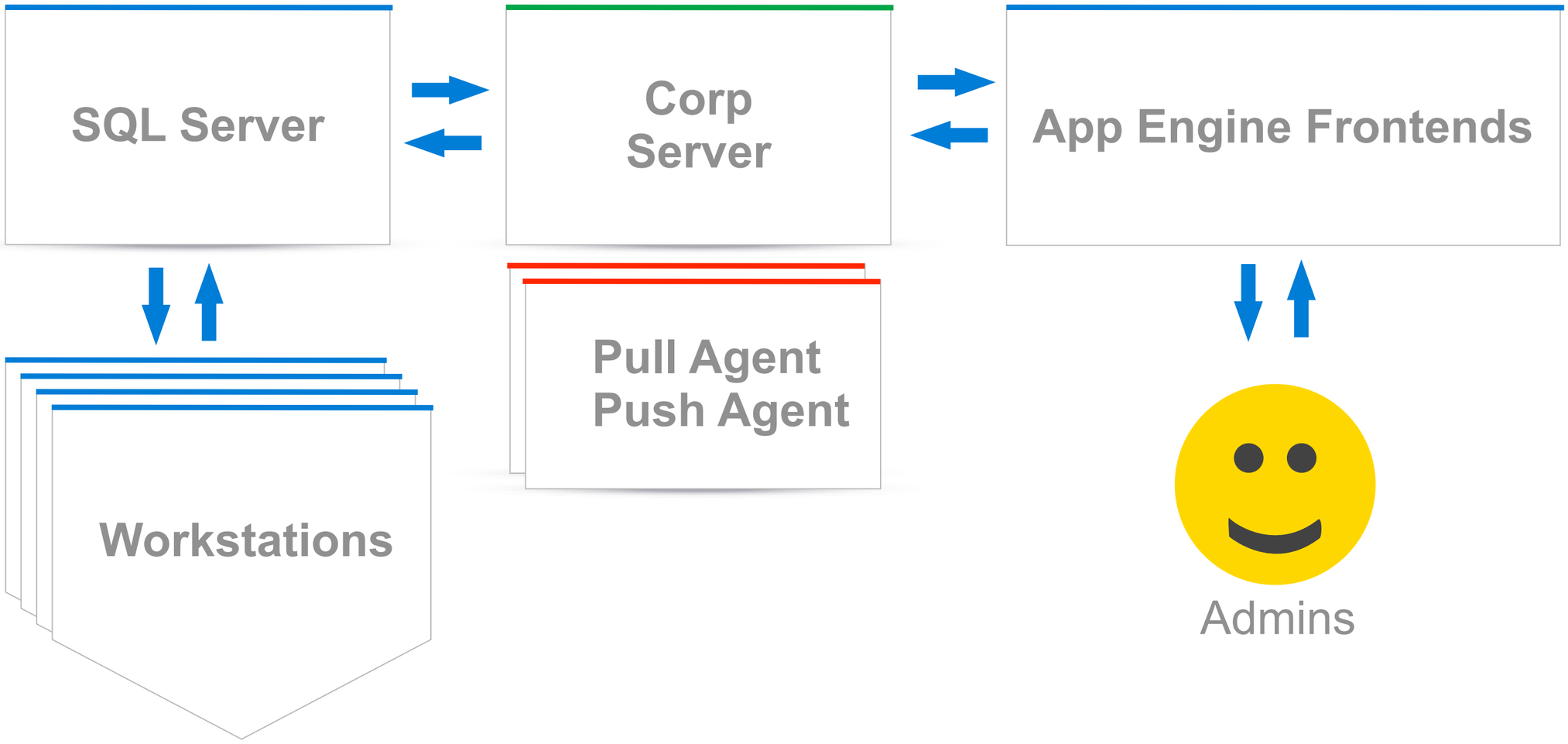
Pattern: Use Agents to Sync with Corp

Integrate existing corporate services with cloud

- Use bulkloader or HTTP PUT/POST to push corporate data in the cloud
- Expose cloud state via HTTP GET endpoints
- Cron or agent client on corporate network polls endpoints for changes
  - Modify data in corporate systems and notify App Engine

Use Agents to Sync with Corp

Use Agents to Sync with Corp



Pattern: Iterative Fetching with Query Cursors

- Respond quickly, process data in background
- Separate work out into small TaskQueue Tasks
- Easily step through queries: UI pagination, checkpointing during large queries, etc
- Portable: cursors can be expressed as strings
- Persistent: continue queries across requests

Simian: Install Reports Cache

Simian: Install Reports Cache

```
def GenerateInstallStats():  # called from cron
```

Simian: Install Reports Cache

```
def GenerateInstallStats():  # called from cron

  install_stats = models.ReportsCache.GetInstallStats()
```

Simian: Install Reports Cache

```
def GenerateInstallStats():  # called from cron
  install_stats = models.ReportsCache.GetInstallStats()
  query = models.InstallLog.all().order('mtime')
```

Simian: Install Reports Cache

```
def GenerateInstallStats():  # called from cron
  install_stats = models.ReportsCache.GetInstallStats()
  query = models.InstallLog.all().order('mtime')
  cursor = models.ReportsCache.GetInstallStatsCursor()
```

Simian: Install Reports Cache

```
def GenerateInstallStats():  # called from cron

  install_stats = models.ReportsCache.GetInstallStats()

  query = models.InstallLog.all().order('mtime')

  cursor = models.ReportsCache.GetInstallStatsCursor()
  if cursor:
    query.with_cursor(cursor)
```

Simian: Install Reports Cache

```
def GenerateInstallStats():  # called from cron

  install_stats = models.ReportsCache.GetInstallStats()

  query = models.InstallLog.all().order('mtime')

  cursor = models.ReportsCache.GetInstallStatsCursor()
  if cursor:
    query.with_cursor(cursor)

  installs = query.fetch(1000)
  if not installs:
    return
```

Simian: Install Reports Cache

```
def GenerateInstallStats():  # called from cron

  install_stats = models.ReportsCache.GetInstallStats()

  query = models.InstallLog.all().order('mtime')

  cursor = models.ReportsCache.GetInstallStatsCursor()
  if cursor:
    query.with_cursor(cursor)

  installs = query.fetch(1000)
  if not installs:
    return

  for install in installs:
    # update install_stats dictionary with various install stats.
    ...
```

Simian: Install Reports Cache

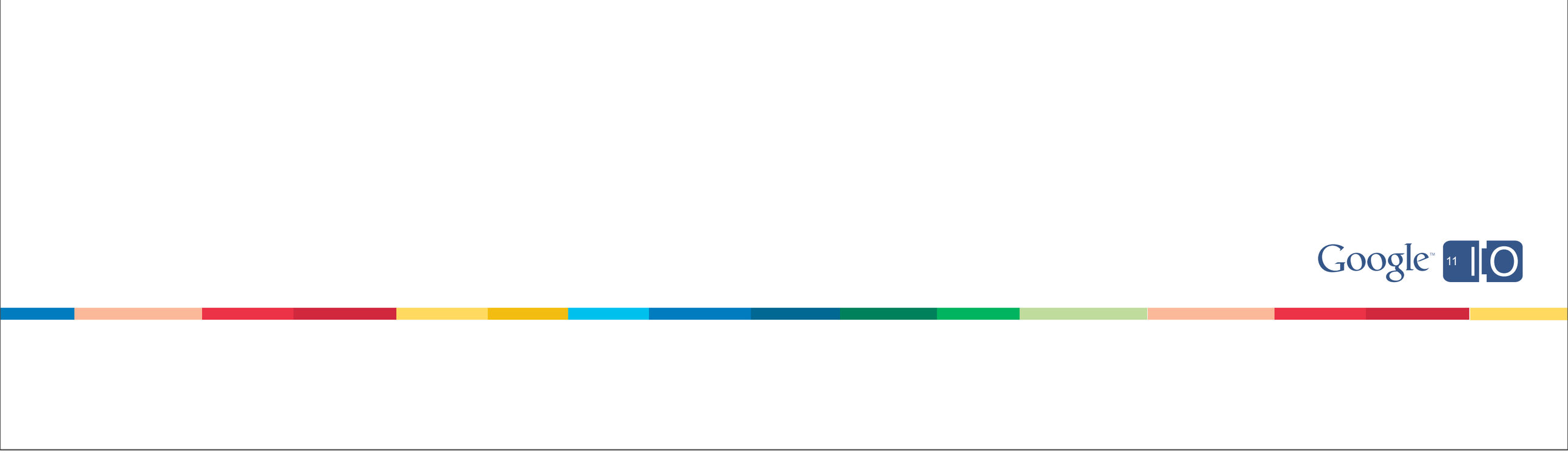
```
def GenerateInstallStats():  # called from cron

  install_stats = models.ReportsCache.GetInstallStats()

  query = models.InstallLog.all().order('mtime')

  cursor = models.ReportsCache.GetInstallStatsCursor()
  if cursor:
    query.with_cursor(cursor)

  installs = query.fetch(1000)
  if not installs:
    return

  for install in installs:
    # update install_stats dictionary with various install stats.
    ...

  models.ReportsCache.SetInstallStats(install_stats)
```

Simian: Install Reports Cache

```
def GenerateInstallStats():  # called from cron

  install_stats = models.ReportsCache.GetInstallStats()

  query = models.InstallLog.all().order('mtime')

  cursor = models.ReportsCache.GetInstallStatsCursor()
  if cursor:
    query.with_cursor(cursor)

  installs = query.fetch(1000)
  if not installs:
    return

  for install in installs:
    # update install_stats dictionary with various install stats.
    ...

  models.ReportsCache.SetInstallStats(install_stats)

  models.ReportsCache.SetInstallStatsCursor(query.cursor())
```

Simian: Install Reports Cache

```
def GenerateInstallStats():  # called from cron

  install_stats = models.ReportsCache.GetInstallStats()

  query = models.InstallLog.all().order('mtime')

  cursor = models.ReportsCache.GetInstallStatsCursor()
  if cursor:
    query.with_cursor(cursor)

  installs = query.fetch(1000)
  if not installs:
    return

  for install in installs:
    # update install_stats dictionary with various install stats.
    ...

  models.ReportsCache.SetInstallStats(install_stats)

  models.ReportsCache.SetInstallStatsCursor(query.cursor())

  deferred.defer(GenerateInstallStats)
```


Google
11

Eric Schoeffler

Staffing Applications

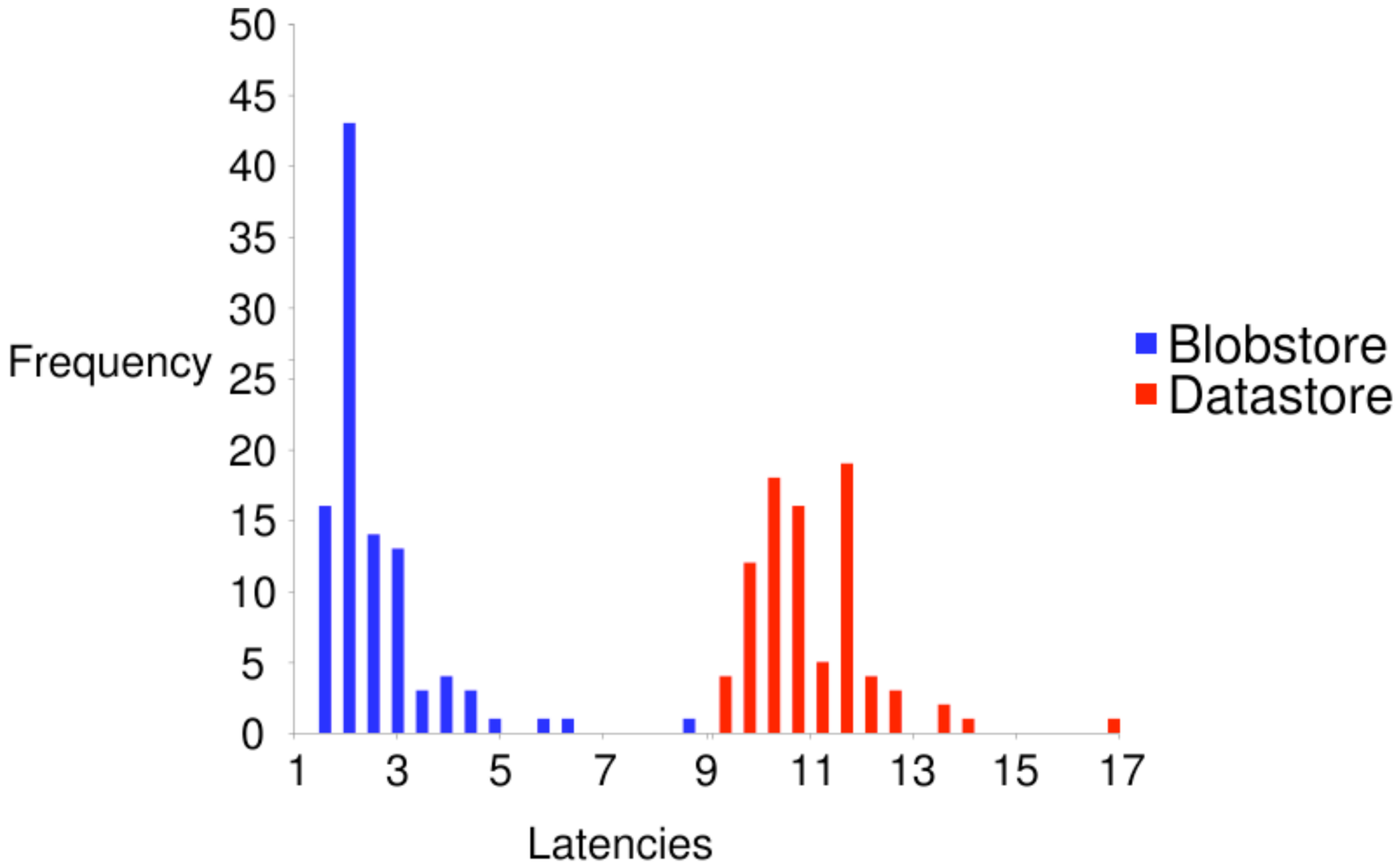
Pattern: Blobstore as Alternate Datasource

- Datastore
  - Not meant for large table scans
  - Limited by query language
    - Inequality filter constraints

- Blobstore
  - Insert / Retrieve / Edit, in bulk
  - Flexible
    - Direct access to Blob data in memory
  - Fast access to Blob data
    - 5MB in ~2s

Demo

100 Example Executions



Step 1: Create Two Servlets on App Engine

a. '/upload' Servlet: returns one time use upload URL, specifies callback

```
String uploadUrl = blobstoreService.createUploadUrl("/postUpload");
```

Step 1: Create Two Servlets on App Engine

a. '/upload' Servlet: returns one time use upload URL, specifies callback

```
String uploadUrl = blobstoreService.createUploadUrl("/postUpload");
```

b. '/postUpload' callback Servlet: stores blob metadata

```
// Called after upload, “req” contains metadata

Map<String, BlobKey> blobKeys = blobstoreService.getUploadedBlobs(req);

// Store blobKey for your blob datastore
```

Step 2: Upload Corporate Data

a. Create a file of serialized records

b. GET upload URL from '/upload'

c. Upload the file

```
curl = pycurl.Curl()
curl.setopt(pycurl.POST, 1)
curl.setopt(pycurl.URL, url)
curl.setopt(
    pycurl.HTTPPOST,
    [('file', (pycurl.FORM_FILE, filename))])
curl.perform()
```

Step 3: Reading the Blob

```
BlobstoreInputStream stream = new BlobstoreInputStream(blobKey);

InputStreamReader reader = new InputStreamReader(stream);

BufferedReader bufferedReader = new BufferedReader(reader);

// Deserialize bufferedReader and run report
```

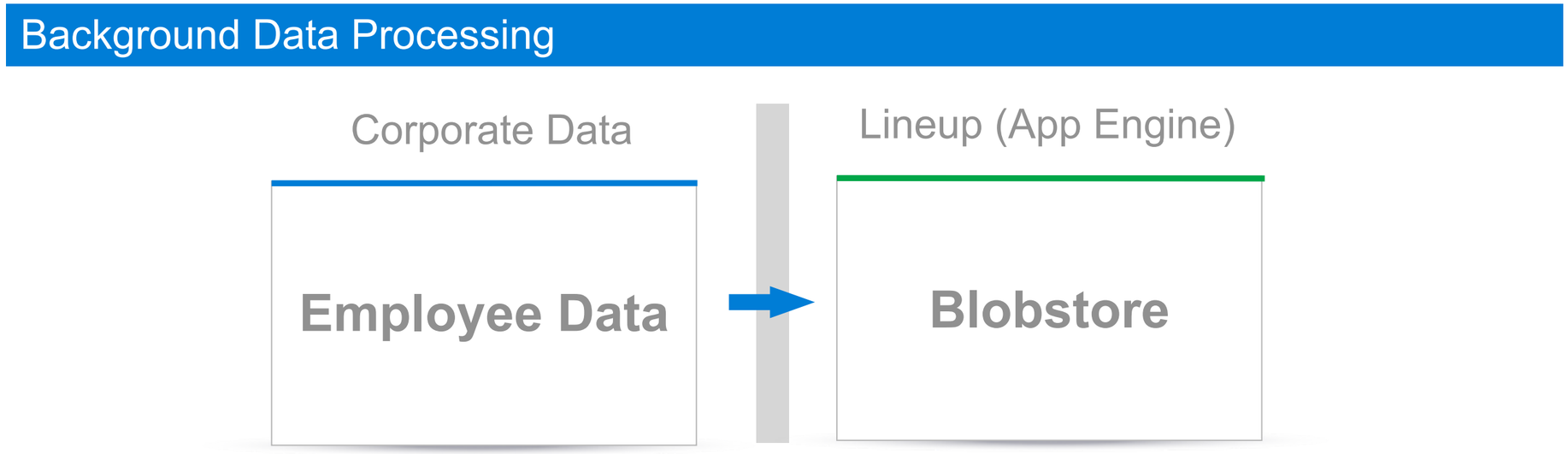
Arbitrary O(n) operations = Flexibility

Example: Lineup

Interview Scheduling

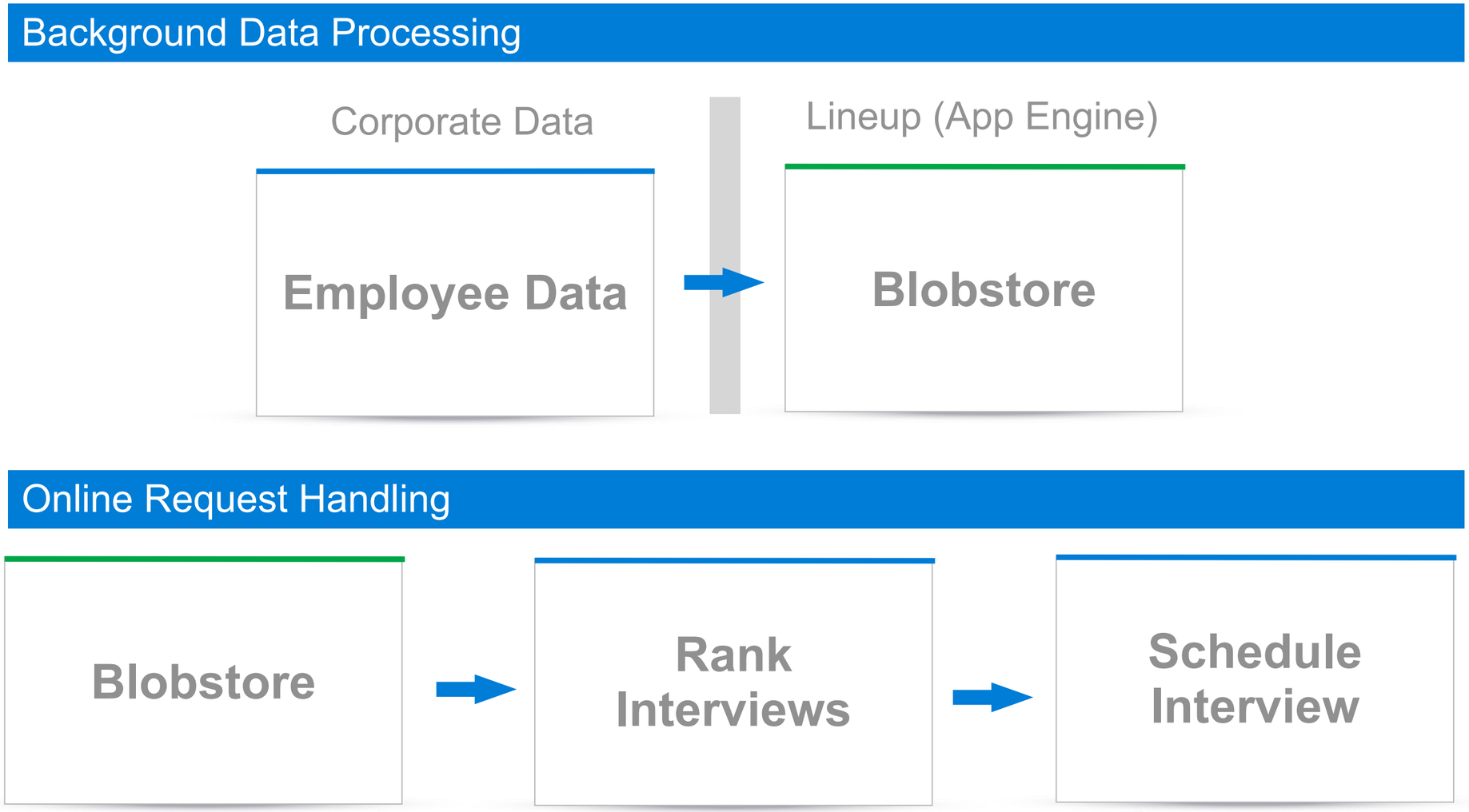
Example: Lineup

Interview Scheduling



Example: Lineup

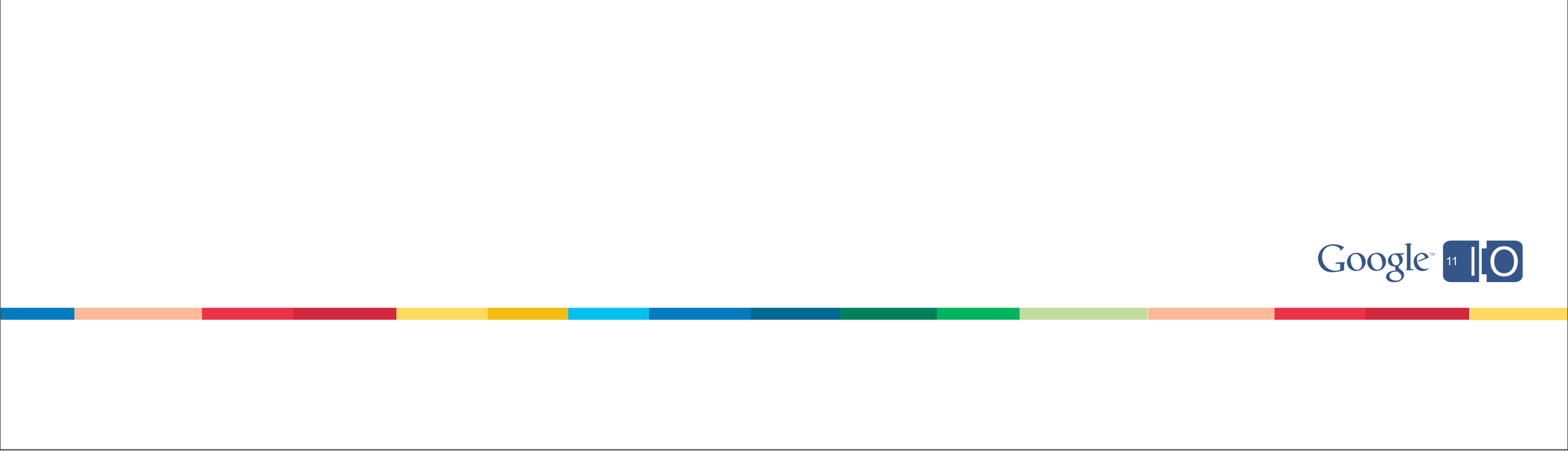
Interview Scheduling



Bottom Line

- Datastore
  - Fetching a few large records
  - Updating individually
  - Scalable to large database size

- Blobstore
  - Fetching many small records
  - Updating all at once
  - Flexible


Google
11

Justin Fagnani

Corporate Engineering

Pattern: Channel API & GWT to Build Collaborative Editing Tool

- Real-time collaborative editing
  - Command Pattern
  - GAE Channel API
  - GWT serialization

- Today we're just talking about how to put the pieces together

Collaboration relies on messages from server to client

- Tricky to do well
- Impossible on App Engine without the Channel API
- Channel API reduces technical complexity

Filling the Gaps

Key Collaboration Responsibilities

1. Manage list of active clients and the objects they're editing

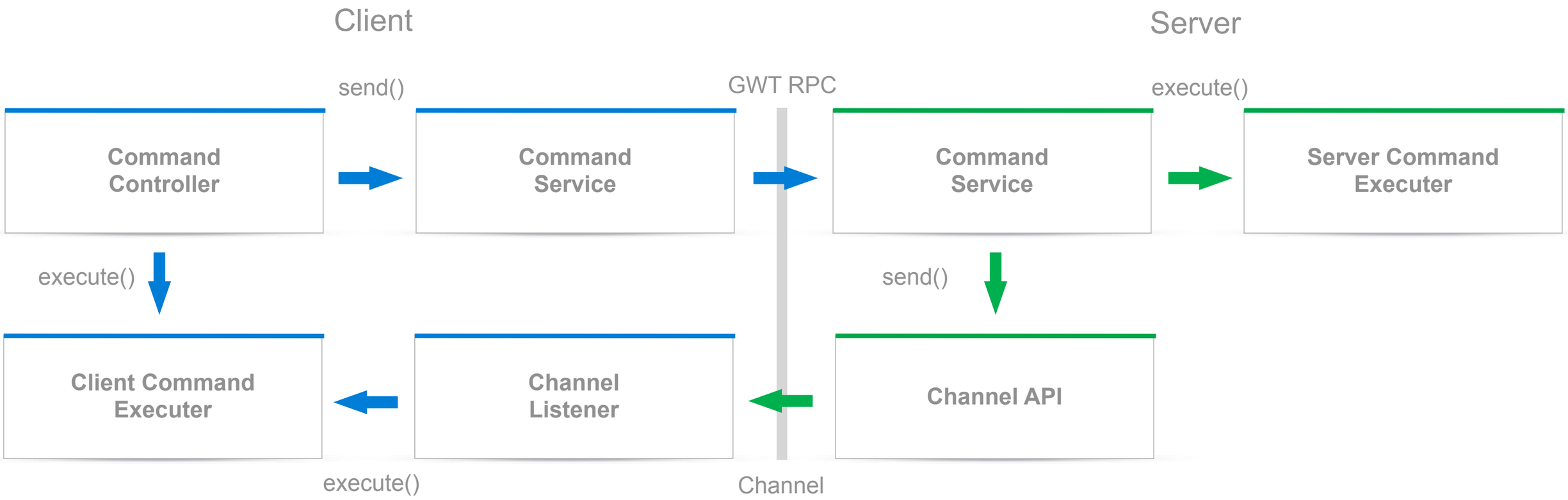
2. Serialize and Deserialize Command objects

3. Execute Commands and dispatch to clients

Main Components

- **Command**
  - Perform action on a given object
- **CommandController**
  - Register client with server, periodically ping
  - Send and receive Command to and from server
- **CommandService**
  - Maintain registry of clients
  - Send and receive Command to and from clients
- **CommandExecutor**
  - Load objects needed by Command
  - Environment-specific (client vs. server) behavior

Main Components



Maintaining Channels

The Channel API doesn't help you track client/object relationships; do it yourself

1. CommandController opens a channel for a specific objectId

2. CommandService stores object->[(clientId,timestamp)…] in memcache

3. CommandController periodically calls CommandService.keepAlive(clientId, objectId)

4. CommandService makes sure it knows about the client, and either:
   - updates the timestamp and sends an OK
   - sends ERROR

5. If ERROR, CommandController re-opens channel and re-syncs the object

Maintaining Channels

Maintaining Channels

```
private String openChannel(String objId, String clientId) {
  boolean succeeded = false;
  String token = channelService.createChannel(clientId);

  for (int tryCount = 0; (tryCount < 5) && !succeeded; tryCount++) {
    IdentifiableValue objChannelsCachedValue = memcache.getIdentifiable(getChannelCacheKey(objId));
```

Maintaining Channels

```
private String openChannel(String objId, String clientId) {
  boolean succeeded = false;
  String token = channelService.createChannel(clientId);

  for (int tryCount = 0; (tryCount < 5) && !succeeded; tryCount++) {
    IdentifiableValue objChannelsCachedValue = memcache.getIdentifiable(getChannelCacheKey(objId));
    Map<String, Long> objChannels = (Map<String, Long>) appChannelsCachedValue.getValue();
    objChannels.put(clientId, clock.now().getMillis());
```

Maintaining Channels

```
private String openChannel(String objId, String clientId) {
  boolean succeeded = false;
  String token = channelService.createChannel(clientId);

  for (int tryCount = 0; (tryCount < 5) && !succeeded; tryCount++) {
    IdentifiableValue objChannelsCachedValue = memcache.getIdentifiable(getChannelCacheKey(objId));
    Map<String, Long> objChannels = (Map<String, Long>) appChannelsCachedValue.getValue();
    objChannels.put(clientId, clock.now().getMillis());
    succeeded = memcache.putIfUntouched(
        getChannelCacheKey(objId), objChannelsCachedValue, objChannels);
  }

  return token;
}
```

Executing a Command

Send a command to the server, apply it and broadcast to listening clients:

1. User performs an action

2. Process user action into Command, send it to CommandController

3. CommandController calls CommandService.executeCommand()

4. CommandService executes Command, sends resulting update Command to listening clients

5. CommandController on listening clients receives the Command and executes it locally

Executing a Command

Executing a Command

```
public Command<?> executeCommand(Command<?> command, String clientId) {
  Command<?> updateCommand = executor.execute(command);
```

Executing a Command

```
public Command<?> executeCommand(Command<?> command, String clientId) {
  Command<?> updateCommand = executor.execute(command);
  String commandMessage = serializer.serializeCommand(updateCommand);
```

Executing a Command

```
public Command<?> executeCommand(Command<?> command, String clientId) {
  Command<?> updateCommand = executor.execute(command);
  String commandMessage = serializer.serializeCommand(updateCommand);
  Map<String, Long> channelIds = getChannelIds(command.getObjectId());
```

Executing a Command

```
public Command<?> executeCommand(Command<?> command, String clientId) {
  Command<?> updateCommand = executor.execute(command);
  String commandMessage = serializer.serializeCommand(updateCommand);
  Map<String, Long> channelIds = getChannelIds(command.getObjectId());
  for (Map.Entry<String, Long> entry : channelIds.entrySet()) {
    String id = entry.getKey();
    if (!id.equals(clientId)) {
      channelService.sendMessage(new ChannelMessage(id, commandMessage));
    }
  }
  return updateCommand;
}
```

GWT Serialization Over Channels

Normally only for RPC calls, but can work with Strings and the Channel API

Server

1. Create a dummy method forcing inclusion of Command in the GWT serialization policy

```
interface CommandService {
  ...
  Command<?> dummyMethod();
}
```

2. Locate RPC manifest, load the serialization policy file

3. Call RPC.encodeResponseForSuccess()

GWT Serialization Over Channels

Normally only for RPC calls, but can work with Strings and the Channel API

Client

1. Cast CommandServiceAsync to SerializationStreamFactory

2. Call streamFactory.createStreamReader(message)

3. Read command with reader.readObject()

GWT Serialization Over Channels

GWT Serialization Over Channels

```
class CommandSerializer {
  String serializeCommand(Command<?> command) {
    Method dummyMethod = CommandService.class.getDeclaredMethod("dummyMethod");
    InputStream in = new FileInputStream(new File(getStrongName()));
    SerializationPolicy policy = SerializationPolicyLoader.loadFromStream
        in, exceptions);
    return RPC.encodeResponseForSuccess(dummyMethod, command, policy);
  }
}
```

GWT Serialization Over Channels

```
class CommandSerializer {
  String serializeCommand(Command<?> command) {
    Method dummyMethod = CommandService.class.getDeclaredMethod("dummyMethod");
    InputStream in = new FileInputStream(new File(getStrongName()));
    SerializationPolicy policy = SerializationPolicyLoader.loadFromStream
        in, exceptions);
    return RPC.encodeResponseForSuccess(dummyMethod, command, policy);
  }
}
```

```
class CommandDeserializer {
  Command<?> deserializeCommand(String message) throws SerializationException {
    SerializationStreamFactory serializationStreamFactory =
        (SerializationStreamFactory) commandServiceAsync;
```

GWT Serialization Over Channels

```
class CommandSerializer {
  String serializeCommand(Command<?> command) {
    Method dummyMethod = CommandService.class.getDeclaredMethod("dummyMethod");
    InputStream in = new FileInputStream(new File(getStrongName()));
    SerializationPolicy policy = SerializationPolicyLoader.loadFromStream
        in, exceptions);
    return RPC.encodeResponseForSuccess(dummyMethod, command, policy);
  }
}
```

```
class CommandDeserializer {
  Command<?> deserializeCommand(String message) throws SerializationException {
    SerializationStreamFactory serializationStreamFactory =
        (SerializationStreamFactory) commandServiceAsync;
    SerializationStreamReader reader =
        serializationStreamFactory.createStreamReader(message);
```

GWT Serialization Over Channels

```
class CommandSerializer {
  String serializeCommand(Command<?> command) {
    Method dummyMethod = CommandService.class.getDeclaredMethod("dummyMethod");
    InputStream in = new FileInputStream(new File(getStrongName()));
    SerializationPolicy policy = SerializationPolicyLoader.loadFromStream
        in, exceptions);
    return RPC.encodeResponseForSuccess(dummyMethod, command, policy);
  }
}
```

```
class CommandDeserializer {
  Command<?> deserializeCommand(String message) throws SerializationException {
    SerializationStreamFactory serializationStreamFactory =
        (SerializationStreamFactory) commandServiceAsync;
    SerializationStreamReader reader =
        serializationStreamFactory.createStreamReader(message);
    return (Command<?>) reader.readObject();
  }
}
```

Collaboration Summary

1. Manage list of active clients and the objects they're editing
    - Map clients to objects in memcache
2. Serialize and Deserialize Command objects
    - Use dummy method
3. Execute Commands and dispatch to clients
    - Send serialized Command via Channel API to listening clients

Wrap Up

- App Engine as a Web Service Platform
- Use Agents to Sync with Corp
- Iterative Fetching with Query Cursors
- Blobstore as an Alternate Datasource
- Channel API & GWT to Build Collaborative Editing Tool

Q&A

Hashtags: #io2011 #AppEngine #cloud #development

Session feedback: http://goo.gl/Td6HC

Reference Code:

Text

http://code.google.com/p/corpengpatterns

http://code.google.com/p/simian


Google™
11
I/O