View live notes and ask questions about this session on Google Wave

http://bit.ly/ds6t9F

Google™ 10 I O

Batch Data Processing with Google App Engine

Mike Aizatsky
20 May 2010

View live notes and ask questions about this session on Google Wave

http://bit.ly/ds6t9F

Agenda

- The challenge
- Early batch processing in App Engine
- Batch processing with Task Queues
- Batch processing at Google
- App Engine approach

Batch Data Processing

- Processing thousands of entities is hard on App Engine
- Some examples:
    - schema migration
    - data export
    - report generation

What Makes It Hard?

- App Engine imposes certain restrictions
- Restrictions guarantee automatic scalability

What Makes It Hard?

- 30s request limit
- Transient errors in the system
- Datastore latency and timeouts
- Changing dataset

Early Batch Processing

Early Batch Processing

- Define batch web handler:

```
class BatchHandler(...):
  def get(self):
    self.processNextBatch()
```

- Use a web page with auto-refresh or use curl:

```
while true; curl http://batch_url; done
```

- Trivia: Was actually done by the App Engine team on the day of release

Challenges

- Need an external "driver" computer (may fail too)
- Difficult error handling and recovery
- Slow, inefficient
- Complex state management

Possible Improvements

- Communicate state with the driver
- Sharding/parallel execution
- Use remote api for complex scenarios
- Typical Example: bulkloader from App Engine SDK

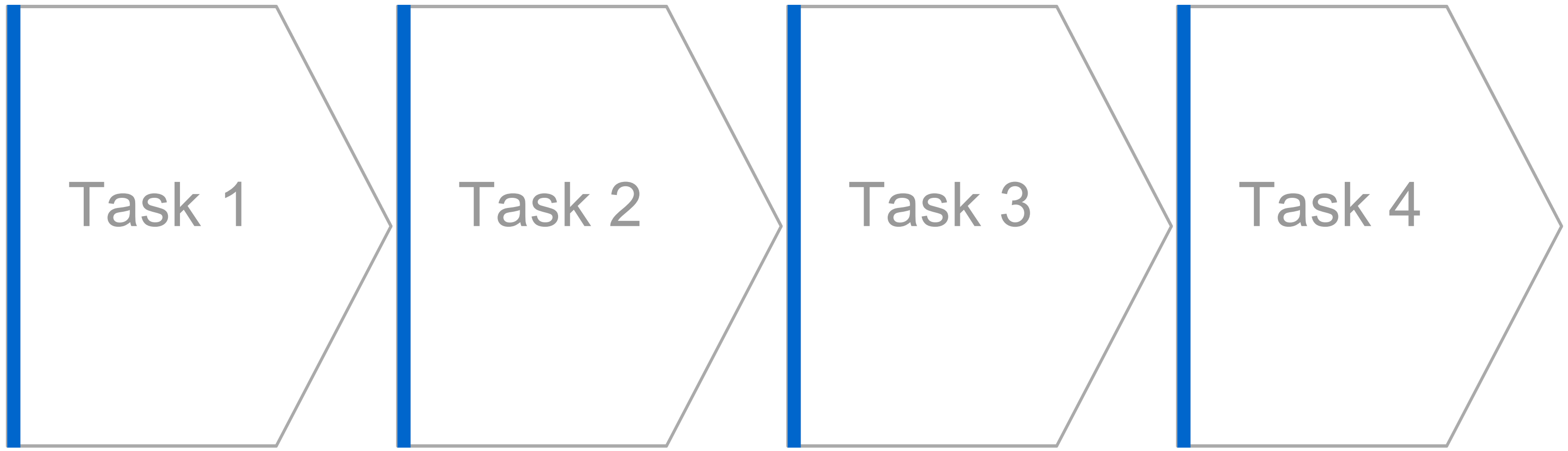
Batch Processing With Task Queues

Batch Processing with Task Queues

- Task Queues released a year ago
- Can perform work outside of a user request
- Reliable, high-performance system
- Still 30s limit

Task Chaining

- Simple technique of overcoming 30s limit
- Task enqueues its continuation when it's close to 30s limit



Task Chaining

- Define batch handler:

```
class BatchHandler(...):
  def get(self):
    next_starting_point =
      self.performNextBatch(
        self.request["starting_point"])
    taskqueue.Task("/batch", params =
     {"starting_point":
      next_starting_point})
      .add("batch_queue")
```

- To start a process, simply enqueue first task

Nice Task Queue Properties

- Guarantees eventual task execution
- No need for external drivers
- Repeats task execution in case of unhandled failures
- Can limit execution rate (both manually and automatically)

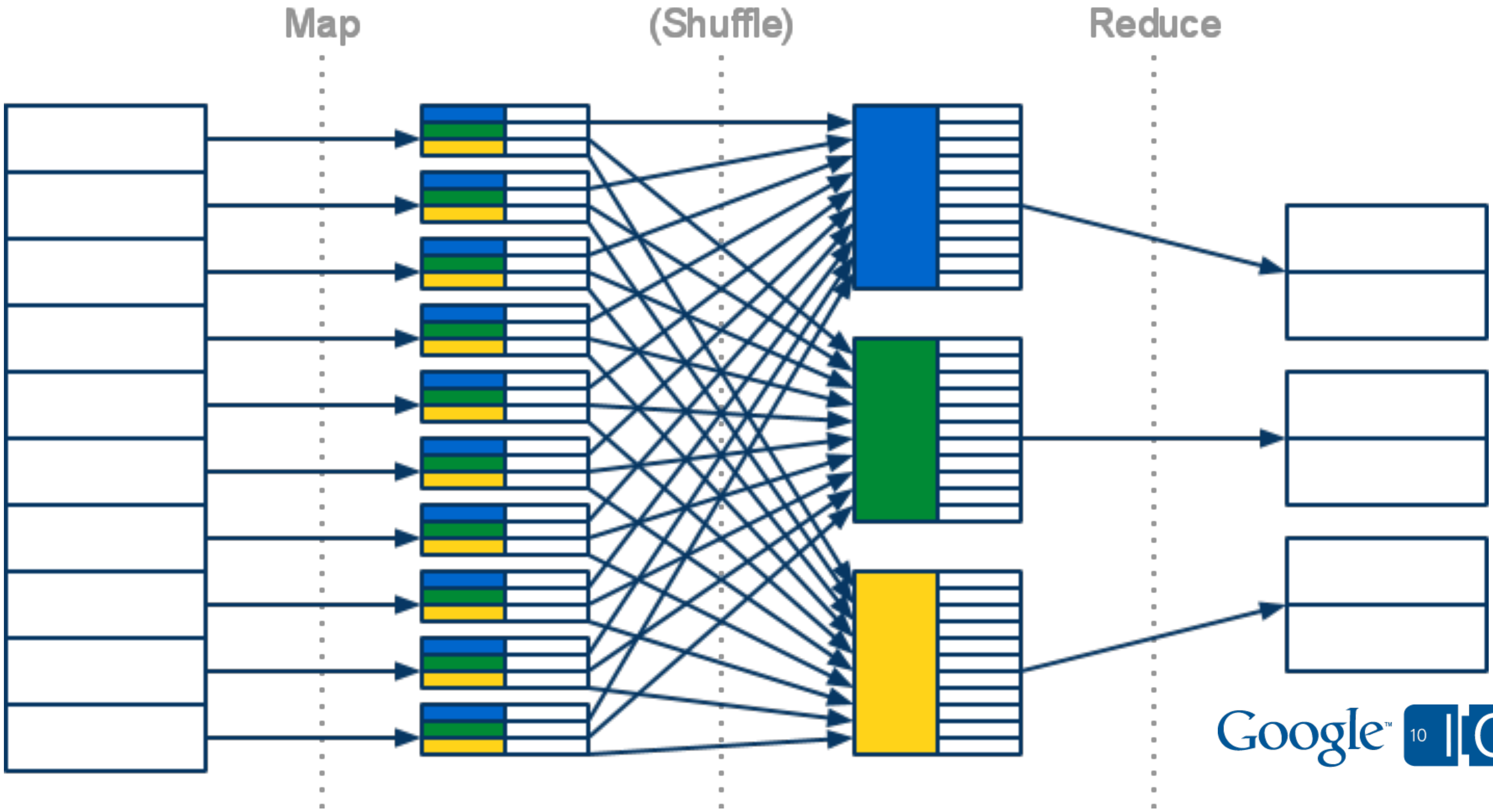
Batch Processing At Google

Batch Processing At Google

- MapReduce successfully used for years to do batch processing at Google scale
- Created to help developers work with unreliable distributed systems
- Widely adopted

MapReduce

- Developer defines only 2 functions:
  - map(entity) -> [(key, value)]
  - reduce(key, [value]) -> [value]



MapReduce Special Cases

- Schema migration: empty reduce, update in map
- Report generation: reduce generates new entities

App Engine & Google's MapReduce

- We want you to use MapReduce too!
- There are some unique challenges

App Engine & Google's MapReduce

- Additional scaling dimension:
  - Lots and lots of applications
  - Many of them will run MapReduce at the same time
- Isolation: application shouldn't influence performance of the other

App Engine & Google's MapReduce

- Rate limiting: you don't want to burn all day's resources in 15min and kill your online traffic
- Very slow execution: free apps want to go really slow, staying under their resource limint
- Protection: from malicious App Engine users

App Engine Approach

App Engine Approach

- We already have a system to solve (most) of these problems: Task Queue
- Decided to build MapReduce on top of Task Queue
- Some additional services will have to be developed

Mapper Library for App Engine

- Early experimental release
- Reliable, fast and efficient way to iterate over datastore or blob files
- Part 1 of the MapReduce story
- You can start playing with it while we're working on the full MapReduce

http://mapreduce.appspot.com/

Mapper Library Features

- Completely user-space. Just pull into your project.
- OSS (Apache 2.0). Hack, modify, play around. Patches welcome!
- Python today & Java soon
- API is very familiar to Hadoop/Dumbo users

Mapper Library Features

- Automatic sharding for faster execution
- Automatic rate limiting for slow execution
- Status pages
- Counters
- Parameterized mappers
- Batching datastore operations
- Iterating over blob data

Demo

Adding Mapper Library To Your Project

- Checkout library from svn into your project folder
- Add 1 handler to app.yaml:

```
- url: mapreduce(/.*)?
  script: <script_location>/main.py
  admin: true
```

Defining Mapper
Python

- Define map function:

```
def process(entity):
  doSomethingWithEntity(entity)
```

- Register in mapreduce.yaml:

```
mapreduce:
- name: Test Mapper
  mapper:
    input_reader: mapreduce.DatastoreInputReader
    handler: model.Entity
```

- Open http://<your_app>/mapreduce and start your mapper

Defining Mapper
Java

```
public class ExampleMapper extends
    AppEngineMapper<Key, Entity> {
  @Override
  public void map(
    Key k, Entity e, Context ctx) {
      processEntity(e);
  }
}
```

Example: Datastore Operations

```
def user_ages(user):
  migrate_user(user)
  yield op.db.Put(user)
```

Example: Counters

```
def user_ages(user):
  yield op.counter.Increment(
    "age-%d" % user.age)
```

Example: More Complex Reports

```
def orders_total(customer):
  orders = Order.by_customer(customer)
  total = sum_orders(orders)
  report_line = ReportLine(
    report_id, customer, total)
  yield op.db.Put(report_line)
```

Some Implementation Details

- Uses task queue chaining
- 2 types of flows: controller flow & worker flow
- Uses datastore for state storage and communication

Important point

- We handle most of the task chaining complexity
- You should handle only one: idempotence

Idempotence

- $f(f(x)) = f(x)$
- Means: your batch handler should be ready to process the same entity twice
- The **most** important property of batch operation
- You should always think about idempotence

Practical Idempotence: Data Migration

Not Idempotent:

```
def migrate(entity):
  yield op.counters.Increment("updated")
  entity.property =
      compute_property(entity)
  yield op.db.Put(entity)
```

Practical Idempotence: Data Migration

Idempotent:

```
def migrate(entity):
  if entity.property_updated:
    return
  yield op.counters.Increment("updated")
  entity.property =
        compute_property(entity)
  entity.property_updated = True
  yield op.db.Put(entity)
```

Practical Idempotence: Reports

Not Idempotent:

```
def report(customer):
  # .....
  report_line = ReportLine(
    report_id, customer, total)
  yield op.db.Put(report_line)
```

Practical Idempotence: Reports

Idempotent:

```
def report(customer):
  # .....
  key_name = "%s-%s" %
      (report_id, customer.id)
  if ReportLine.get_by_key_name(key_name):
    return
  report_line = ReportLine(
    key_name=key_name,
    report_id, customer, total)
  yield op.db.Put(report_line)
```

Practical Idempotence: Counters

Not Idempotent:

```
def user_ages(user):
  yield op.counter.Increment(
    "age-%d" % user.age)
```

Practical Idempotence: Counters

- No easy way to achieve
- Arguably not needed

Idempotence And Changing Data

- Most reports over live data are approximate
- Approximate reports are OK for most cases
- Margin of error should be quite small due to the way mapper library is implemented

Summary

- Mapper library available today
- Reliable, fast and efficient way to iterate over datastore or blob files
- Java & Python
- Fully OSS

http://mapreduce.appspot.com/

Google™ 10 IO