COMSM0010 Cloud Computing

Lecture 05

*-as-a-Service

Dave Cliff

Department of Computer Science
University of Bristol

csdtc@bristol.ac.uk

University of BRISTOL

Stuff-as-a-Service

Remember from Lecture 1…

— Copyright © 2019, Dave Cliff

A common (and useful) initial stratification: *aaS

- SaaS: Software as a Service
  - End-user application software functionality remotely delivered over the internet/web
  - You used to buy this as a CD/DVD in a shrink-wrapped cardboard box

- PaaS: Platform as a Service
  - Developer application software (middleware) functionality, remotely accessible
  - E.g. provide a particular combination of OS, web-server, data-base, & scripting
  - Popular instance is the free "LAMP Stack" – Linux, Apache, MySQL, and PHP(/Perl/Python)

- IaaS: Infrastructure as a Service
  - IT infrastructure, almost always virtualized, remotely accessible, "bare metal"

— Copyright © 2019, Dave Cliff

A common (and useful) initial stratification: *aaS

- SaaS: Software as a Service
- End-user application software functionality remotely delivered over the internet/web
- You used to buy this as a CD/DVD in a shrink-wrapped cardboard box

- PaaS: Platform as a Service
- Developer application software (middleware) functionality, remotely accessible
- E.g. provide a particular combination of OS, web server, data-base, & scripting
- Popular instance is the free "LAMP Stack" – Linux, Apache, MySQL, and PHP(/Perl/Python)

GOOGLE APP ENGINE

- IaaS: Infrastructure as a Service
- IT infrastructure, almost always virtualized, remotely accessible, "bare metal"

AMAZON EC2, S3, SQS

— Copyright © 2019, Dave Cliff

A common (and useful) initial stratification: *aaS

- SaaS: Software as a Service
  - End-user application software functionality remotely delivered over the internet/web
  - You used to buy this as a CD/DVD in a shrink-wrapped cardboard box

- PaaS: Platform as a Service
  - Developer application software (middleware) functionality, remotely accessible
  - E.g. provide a particular combination of OS, web server, data-base, [illegible] scripting
  - Popular instance is the free "LAMP Stack" – Linux, Apache, MySQL, and [illegible] (Python)

GOOGLE APP ENGINE

- IaaS: Infrastructure as a Service
  - IT infrastructure, almost always virtualized, remotely accessible, "bare metal"

AMAZON EC2, S3, SQS

GOOGLE COMPUTE ENGINE

— Copyright © 2019, Dave Cliff

A common (and useful) initial stratification: *aaS

- SaaS: Software as a Service
- End-user application software functionality remotely delivered over the internet/web
- You used to buy this as a CD/DVD in a shrink-wrapped cardboard box

- PaaS: Platform as a Service
- Developer application software (middleware) functionality, remotely accessible
- E.g. provide a particular combination of OS, web server, data-base, & scripting
- Popular instance is the free "LAMP Stack" – Linux, Apache, MySQL, and PHP(/Perl/Python)

GOOGLE APP ENGINE

- IaaS: Infrastructure as a Service
- IT infrastructure, almost always virtualized, remotely accessible, "bare metal"

AMAZON EC2, S3, SQS

GOOGLE COMPUTE ENGINE

— Copyright © 2019, Dave Cliff

A common (and useful) initial stratification: *aaS

- SaaS: Software as a Service
- End-user application software functionality remotely delivered over the internet/web
- You used to buy this as a CD/DVD in a shrink-wrapped cardboard box

- PaaS: Platform as a Service
- Developer application software (middleware) functionality, remotely accessible
- E.g. provide a particular combination of OS, web server, data-base, & scripting
- Popular instance is the free "LAMP Stack" – Linux, Apache, MySQL, and PHP(/Perl/Python)

GOOGLE APP ENGINE

AMAZON APPSTREAM, DYNAMODB, RDS

- IaaS: Infrastructure as a Service
- IT infrastructure, almost always virtualized, remotely accessible, "bare metal"

AMAZON EC2, S3, SQS

GOOGLE COMPUTE ENGINE

— Copyright © 2019, Dave Cliff

A common (and useful) initial stratification: *aaS

- SaaS: Software as a Service
- End-user application software functionality remotely delivered over the internet/web
- You used to buy this as a CD/DVD in a shrink-wrapped cardboard box

- PaaS: Platform as a Service
- Developer application software (middleware) functionality, remotely accessible
- E.g. provide a particular combination of OS, web server, data-base, & scripting
- Popular instance is the free "LAMP Stack" – Linux, Apache, MySQL, and PHP(/Perl/Python)

GOOGLE APP ENGINE

AMAZON APPSTREAM, DYNAMODB, RDS

- IaaS: Infrastructure as a Service
- IT infrastructure, almost always virtualized, remotely accessible, "bare metal"

AMAZON EC2, S3, SQS

GOOGLE COMPUTE ENGINE

— Copyright © 2019, Dave Cliff

A common (and useful) initial stratification: *aaS

- SaaS: Software as a Service
- End-user application software functionality remotely delivered over the internet/web
- You used to buy this as a CD/DVD in a shrink-wrapped cardboard box

EVERYONE

- PaaS: Platform as a Service
- Developer application software (middleware) functionality, remotely accessible
- E.g. provide a particular combination of OS, web server, data-base, & scripting
- Popular instance is the free "LAMP Stack" – Linux, Apache, MySQL, and PHP(/Perl/Python)

GOOGLE APP ENGINE

AMAZON APPSTREAM, DYNAMODB, RDS

- IaaS: Infrastructure as a Service
- IT infrastructure, almost always virtualized, remotely accessible, "bare metal"

AMAZON EC2, S3, SQS

GOOGLE COMPUTE ENGINE

— Copyright © 2019, Dave Cliff

A common (and useful) initial stratification: *aaS

- SaaS: Software as a Service
- End-user application software functionality remotely delivered over the internet/web
- You used to buy this as a CD/DVD in a shrink-wrapped cardboard box

EVERYONE

REALLY, EVERYONE

- PaaS: Platform as a Service
- Developer application software (middleware) functionality, remotely accessible
- E.g. provide a particular combination of OS, web server, data-base, & scripting
- Popular instance is the free "LAMP Stack" – Linux, Apache, MySQL, and PHP(/Perl/Python)

GOOGLE APP ENGINE

AMAZON APPSTREAM, DYNAMODB, RDS

- IaaS: Infrastructure as a Service
- IT infrastructure, almost always virtualized, remotely accessible, "bare metal"

AMAZON EC2, S3, SQS

GOOGLE COMPUTE ENGINE

— Copyright © 2019, Dave Cliff

and then…

— Copyright © 2019, Dave Cliff

A common (and useful) initial stratification: *aaS

- SaaS: Software as a Service
  - End-user application software fully [illegible] delivered over the internet/web
  - You used to buy this as a CD/DVD in a shrink-wrapped cardboard box

EVERYONE

REALLY, EVERYONE

- PaaS: Platform as a Service
  - Developer application software (middleware) functionality, remotely accessible
  - E.g. provide a particular combination of OS, web-server, data-base, & scripting
  - Popular [illegible] Linux, Apache, MySQL and PHP/Perl/Python)

GOOGLE APP ENGINE

AMAZON APPSTREAM, DYNAMODB, RDS

- IaaS: Infrastructure as a Service
  - IT infrastructure, almost always virtualized [illegible] accessible, "bare metal"

AMAZON EC2, S3, SQS

GOOGLE COMPUTE ENGINE

— Copyright © 2019, Dave Cliff

A common (and useful) initial stratification: *aaS

- FaaS: Function as a Service (AKA "Serverless")
- No server processes visibly running. Pay only for the time spent executing a function.
- Unlike PaaS, scale out without increasing number of servers.
- Amazon Lambda is the best-known example (cf Google Cloud Fns, MSFT Azure Fns).

- SaaS: Software as a Service
- End-user application software fu[illegible] delivered over the internet/web
- You used to buy this as a CD/DVD in a shrink-wrapped cardboard box

EVERYONE

REALLY, EVERYONE

- PaaS: Platform as a Service
- Developer application software (middleware) functionality, remotely accessible
- E.g. provide a particular combination of OS, web-server, data-base, & scripting
- Popu[illegible] Lin[illegible] Perl/Perl/Python)

GOOGLE APP ENGINE

AMAZON APPSTREAM, DYNAMODB, RDS

- IaaS: Infrastructure as a Service
- IT infrastructure, alm[illegible] "bare metal"

AMAZON EC2, S3, SQS

GOOGLE COMPUTE ENGINE

Stuff-as-a-Service

Remember from Lecture 2…

— Copy

IaaS basics via AWS

(Last lecture: broad overview; introduced IaaS/PaaS/SaaS distinction)

IaaS in more detail via the basics of a minimal set of initial AWS services:

- Amazon Simple Storage Service (S3)
- Amazon Elastic Compute Cloud (EC2)
- Amazon Simple Queue Service (SQS)

Stuff-as-a-Service

NOT covered in Lecture 2…

AWS SimpleDB

— Copy

AWS SimpleDB: basics

Provides:

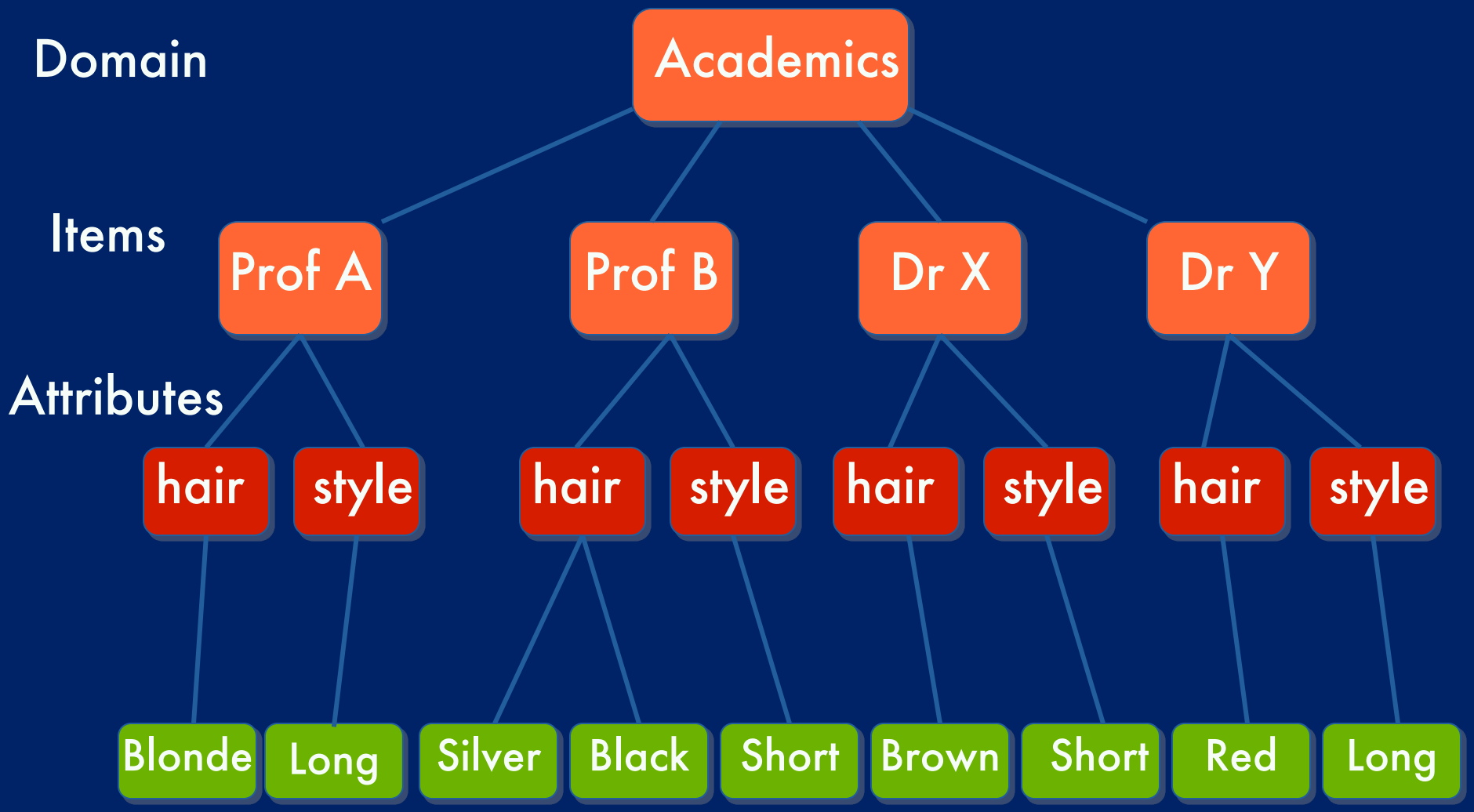
- Reliable storage of structured textual data
- Language that allows you to store, modify, query, & retrieve data sets
- Automatic indexing of all your stored data

SimpleDB provides 3 main resources:

- **Domains:** highest-level container for related data items; queries only search within one domain
- **Items:** a named collection of attributes that represents a data object; each item has a unique name within the domain; items can be created, modified, or deleted; individual attributes within an item can be manipulated
- **Attributes:** an individual category of information within the item, with a name unique for that item; item has one or more text string values associated with the name

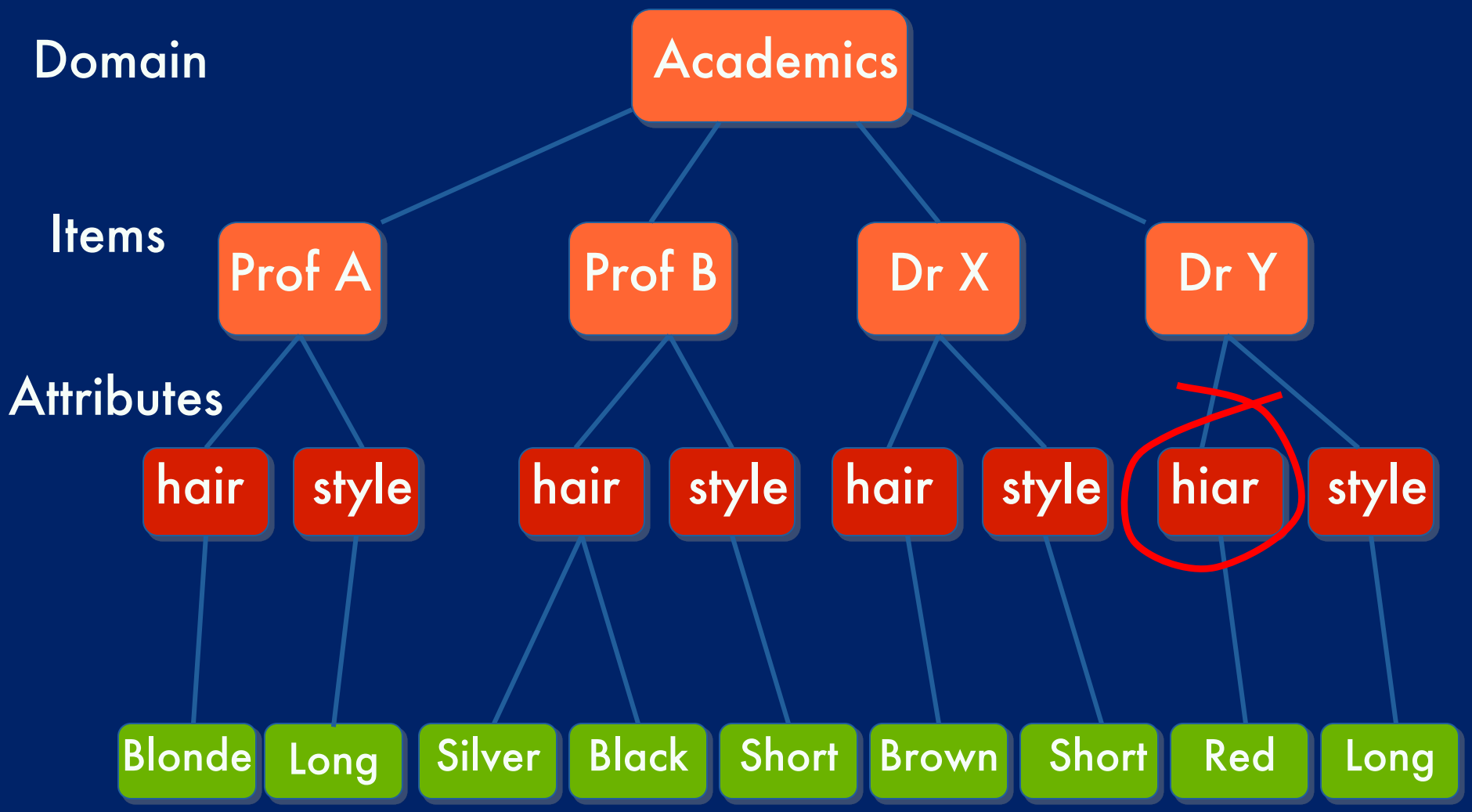
— Copyright © 2014, Dave Cliff

Amazon SimpleDB: an example hierarchy



— Copyright © 2014, Dave Cliff

Amazon SimpleDB: an example hierarchy



— Copy

Amazon SimpleDB: not an RDBMS

NOT a traditional Relational Database Management System (RDBMS) – it is simple

Primary differences:

- SimpleDB Items are stored in a hierarchical structure, not a table
- SimpleDB attribute value max size =1Kb
- SimpleDB data is all stored as text – there are no other data-types!
- SimpleDB's query language is, um, simple in comparison to SQL et al
- SimpleDB is distributed, so data consistency may suffer due to propagation delays

Stuff-as-a-Service

FaaS covered in more depth in Lecture 10 (Dan, 31/10/19)

Today… Focus on PaaS and a bit of CaaS

Stuff-as-a-Service

FaaS covered in more depth in Lecture 10 (Dan, 31/10/19)

Today… Focus on PaaS and a bit of CaaS

What is CaaS?

Container as a Service

Containers coming in L06 & L07

Stuff-as-a-Service

FaaS covered in more depth in Lecture 10 (Dan, 31/10/19)

Today… Focus on PaaS and a bit of CaaS

What is CaaS?

Container as a Service

Containers coming in L06 & L07

Stuff-as-a-Service

FaaS covered in more depth in Lecture 10 (Dan, 31/10/19)

Today… Focus on PaaS and a bit of CaaS

…via the medium of….

Stuff-as-a-Service

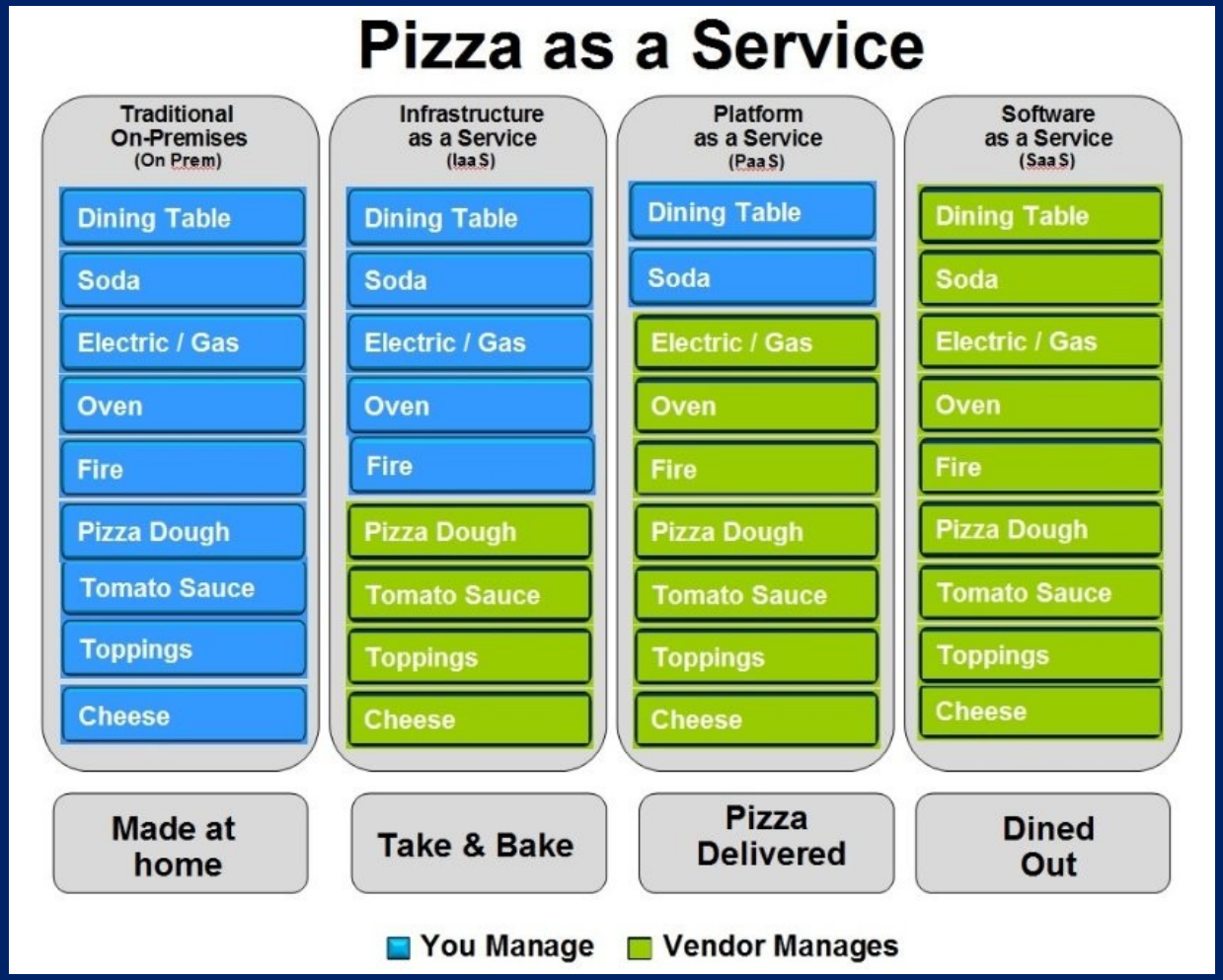
FaaS covered in more depth in Lecture 10 (Dan, 31/10/19)

Today… Focus on PaaS and a bit of CaaS

…via the medium of….

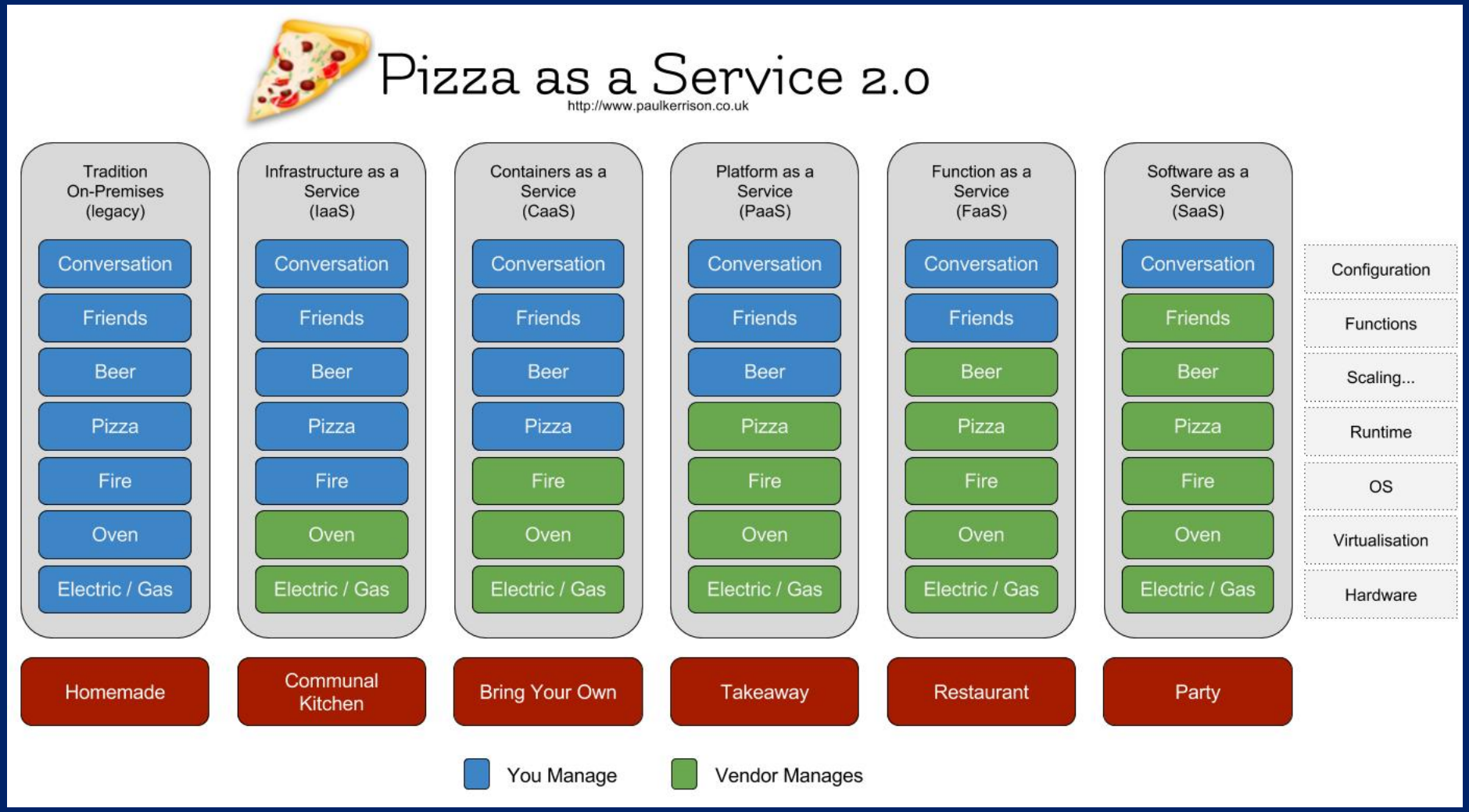
Pizza-as-a-Service 1.0 (Albert Barron @ IBM)

https://www.linkedin.com/pulse/20140730172610-9679881-pizza-as-a-service/



Pizza-as-a-Service 2.0

https://www.paulkerrison.co.uk/random/pizza-as-a-service-2-0

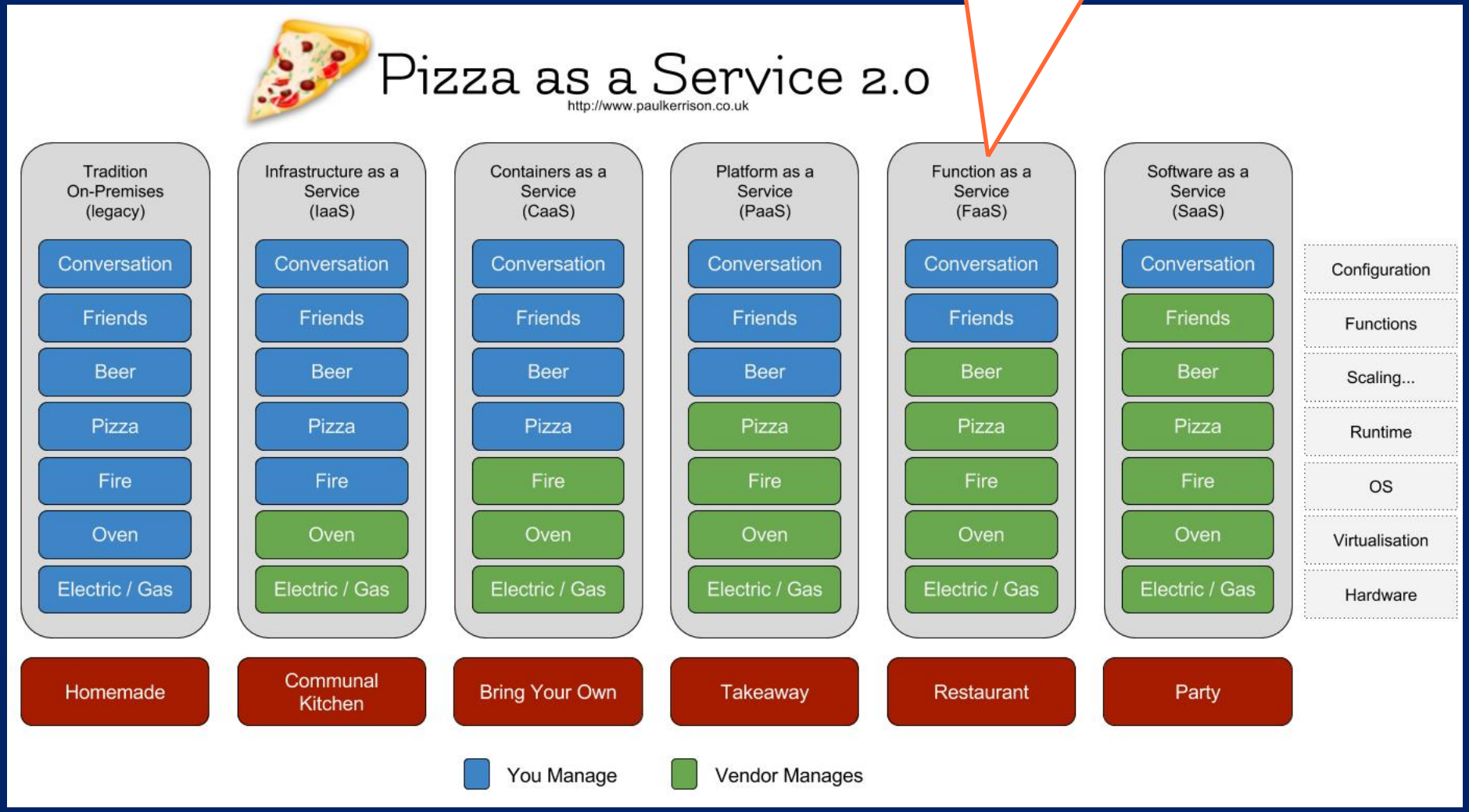


Pizza-as-a-Service 2.0

https://www. -2-0

This feels a bit labored.

FaaS is more like a waiter that loads a small chunk of pizza onto your fork every time you want another bite



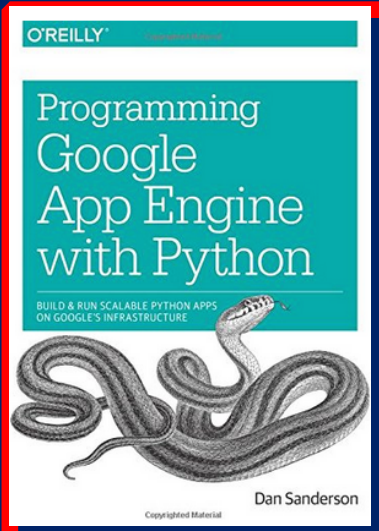
Example PaaS: Google App Engine

Two very useful books (especially Sanderson 2012)

2012 (2/e)

2009

Sanderson 2012 still on sale, but "forked" in 2015



2015

2015

2009

See also https://developers.google.com/appengine/docs/

App Engine > Documentation

Google App Engine Documentation

SEND FEEDBACK

App Engine is a fully managed, serverless platform for developing and hosting web applications at scale. You can choose from several popular languages, libraries, and frameworks to develop your apps, then let App Engine take care of provisioning servers and scaling your app instances based on demand.

Pick a language to learn more.

Go View docs	PHP View docs	Java View docs
Python View docs	Node.js View docs	.NET View docs
Ruby View docs	More languag... View custom runtime docs	

GAE: PaaS

We've seen in earlier lectures that *Infrastructure as a Service* (IaaS) offers access to raw infrastructure: lots of options, user chooses and configures:

- Very powerful, wide range of options and possible configurations
- Often you end up needing to do System-Administrator jobs

Google App Engine is *Platform as a Service* (PaaS): a lot of the low-level administration/management is automated, and abstracted away

- Less power, narrower range of choices
- Less hassle: the automation lets you concentrate on the key stuff
- Automated scaling out of databases, automated distributed computation
- App engine scalable datastore does not behave like a traditional relational DB

Programming GAE: Python, Java, Go, PHP

GAE supports three programming "technology stacks"

- Python (originally this was the *only* language supported)
  - Interpreted, so can be slow; but works with web app frameworks like Django
- Java (and, via JVM, other languages like Clojure, Scala, Ruby, etc)
- More recently: PHP & Go (Google's own language)
  - Go compiles on the GAE server and hence executes at native CPU speeds

Fundamentally, GAE is a web application hosting service.

Enables the construction of scalable apps/services accessed via the Web

You can use GAE for traditional static (documents+images) websites

But it's been designed for real-time dynamic applications

- E.g. multiplayer games, social networks, stores with shopping carts, etc

Programming GAE: Python, Java, Go, PHP

2015

GAE supports three programming "technology stacks"

- Python (originally this was the *only* language supported)
  - Interpreted, so can be slow; but works with web app frameworks like Django
- Java (and, via JVM, other languages like Clojure, Scala, Ruby, etc)
- More recently: PHP & Go (Google's own language)
  - Go compiles on the GAE server and hence executes at native CPU speeds

Fundamentally, GAE is a web application hosting service.

Enables the construction of scalable apps/services accessed via the Web

You can use GAE for traditional static (documents+images) websites

But it's been designed for real-time dynamic applications

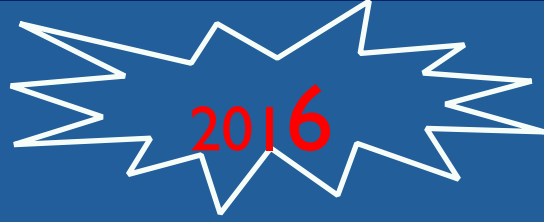
- E.g. multiplayer games, social networks, stores with shopping carts, etc

Programming GAE: Python, Java, etc

GAE supports three programming "technology stacks"

- Python (originally this was the *only* language supported)
  - Interpreted, so can be slow; but works with web app frameworks like Django
- Java (and, via JVM, other languages like Clojure, Scala, Ruby, etc)
- More recently: PHP & Go then Node & Ruby
  - Go compiles on the GAE server and hence executes at native CPU speeds

Programming GAE: Python, Java, etc



GAE supports three programming "technology stacks"

- Python (originally this was the *only* language supported)
  - Interpreted, so can be slow; but works with web app frameworks like Django
- Java (and, via JVM, other languages like Clojure, Scala, Ruby, etc)
- More recently: PHP & Go then Node & Ruby
  - Go compiles on the GAE server and hence executes at native CPU speeds

Python, Java, PHP, Go all available in the "Standard Environment"

Google's custom *containers*

This is what GAE has used for ever

Programming GAE: Python, Java, etc

2016

Source: Sanderson (2013) Ch1.

GAE supports three programming "technology stacks"

- Python (originally this was the *only* language supported)
  - Interpreted, so can be slow; but works with web app frameworks like Django
- Java (and, via JVM, other languages like Clojure, Scala, Ruby, etc)
- 1[illegible] recently: PHP & Go then Node & Ruby
  - [illegible]mpiles on the GA[illegible] server and hence executes at native CPU speeds

Python, Java, Go, Node, Ruby (*not* PHP)
in the "Flexible Environment"

Docker *containers*
New since Aug'16
see Lectures 6 & 7

Costs (https://cloud.google.com/pricing/, Oct 2013)

As with AWS, GAE bills you for metered usage per hour/month.

You only pay for what you use, and as with AWS, there is a free tier:

All applications have a free amount of usage (reset daily) and Paid/Premier applications can use more at the prices below

	Free quota per app per day	Pricing if you exceed your free quota
Hosting	Free quota per app per day	Price
On-demand Frontend Instances	28 free instance hours	$0.08 / hour
Reserved Frontend Instances		$0.05 / hour
High Replication Datastore	1G	$0.18 / GB / month
Outgoing Bandwidth	1G	$0.12 / GB
Incoming Bandwidth	1G	Free
APIs		
Datastore API	50k free read/write/small	$0.09/100k write ops $0.06/100k read ops $0.01/100k small ops

GAE Costs, Oct 2014

Services

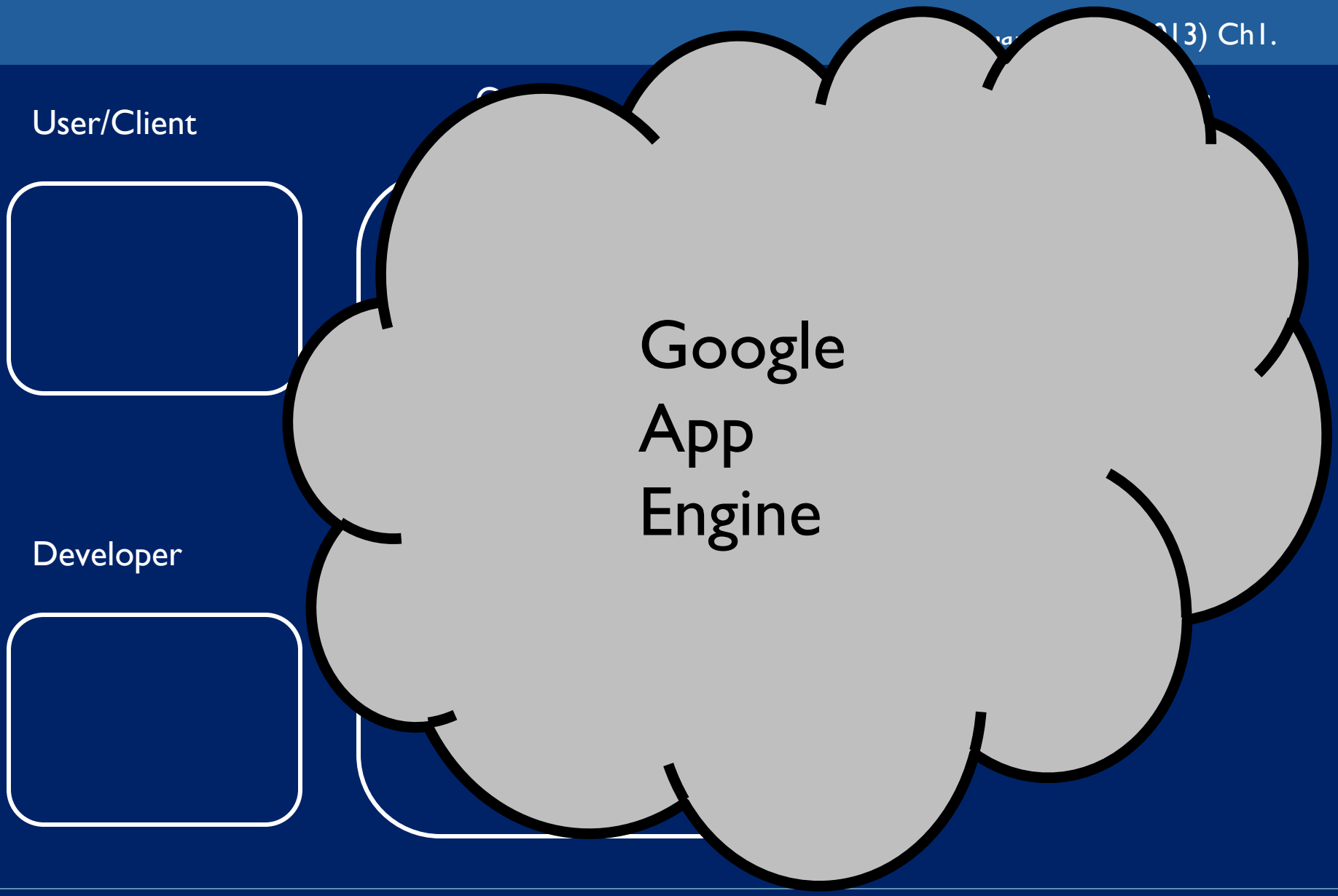
All applications are free within a usage quota that is reset daily. Paid applications can use more at the prices below.

Service	Free quota per app per day		Pricing if you exceed your free quota
Instances	28 instance hours	-38%	$0.05 / instance / hour
Cloud Datastore (NoSQL Database)	50k read/write/small 1 GB storage	0%	$0.06 / 100k read or write ops Small operations free* $0.18 / GB / month
Outgoing Network Traffic	1 GB	0%	$0.12 / GB
Incoming Network Traffic	1 GB		Free
Cloud Storage	5 GB	-80%	$0.026 / GB / month
Memcache	Free Usage of Shared Pool No free quota for Dedicated Pool		Free Usage of Shared Pool Dedicated Pool: $0.06 / GB / hour
Search	1000 basic operations 0.01 GB indexing documents 0.25 GB document storage 100 searches	+500% 0% 0%	$0.50 / 10k searches ~~$2.00 / GB indexing documents~~ $0.18 / GB / month Storage
Email API	100 recipients		Contact Sales
Logs API	100 MB		$0.12 per GB
Task Queue and Logs Storage	5 GB 1 GB		$0.026 / GB / month

GAE Costs, Oct 2015 (and ever since)

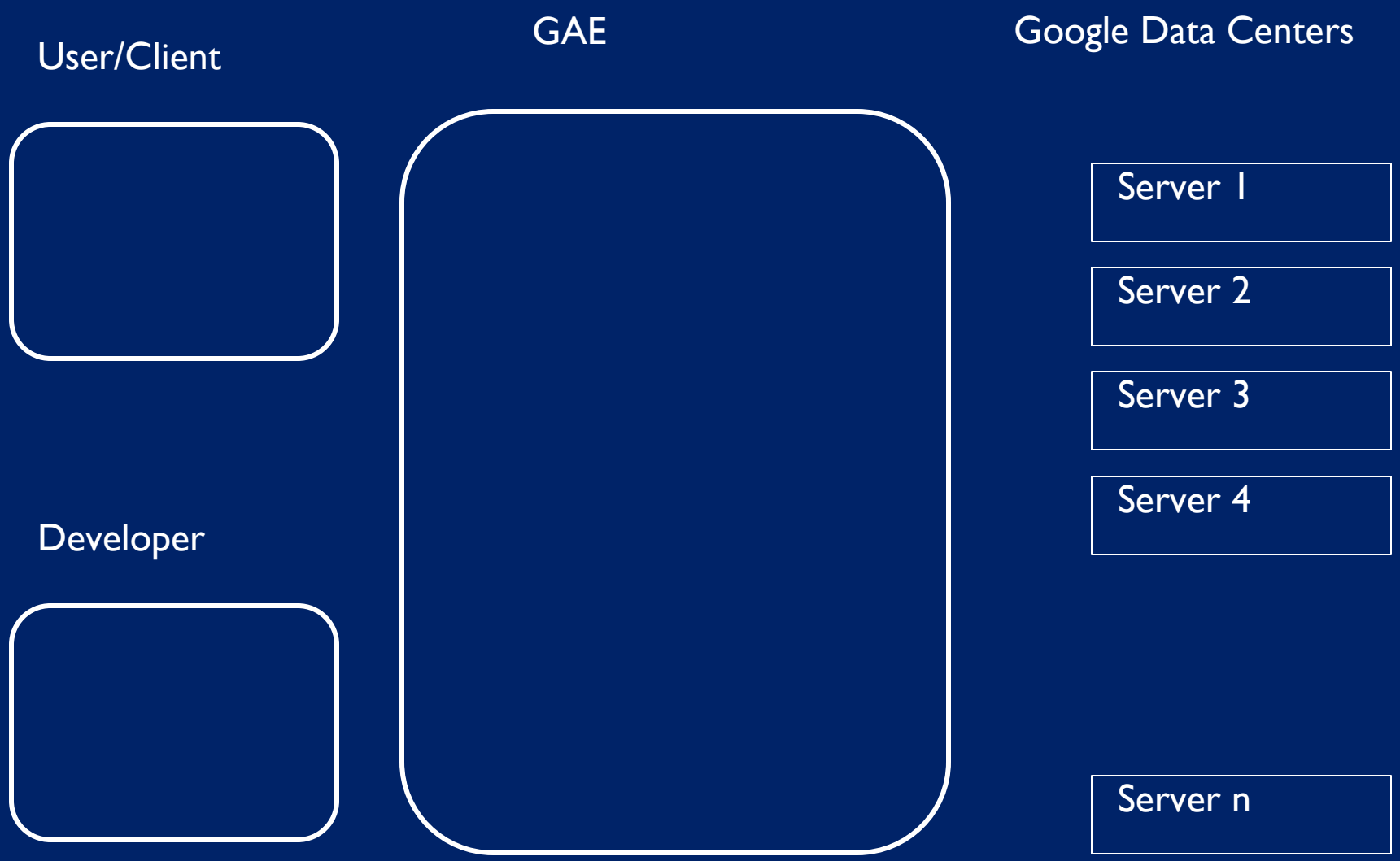
Enter details into the online calculator…

GAE Runtime Environment



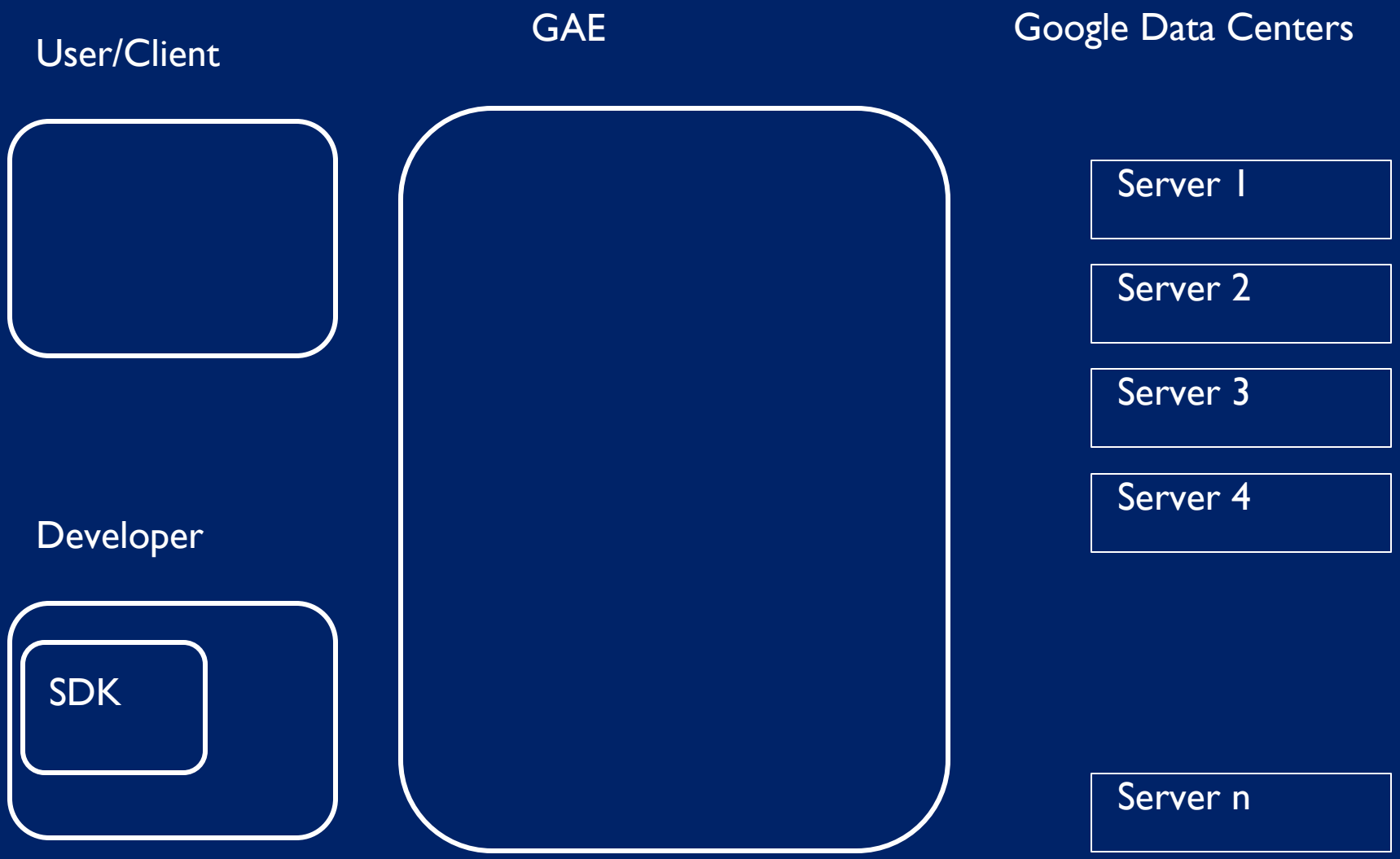
GAE Runtime Environment

Source: Sanderson (2013) Ch1.



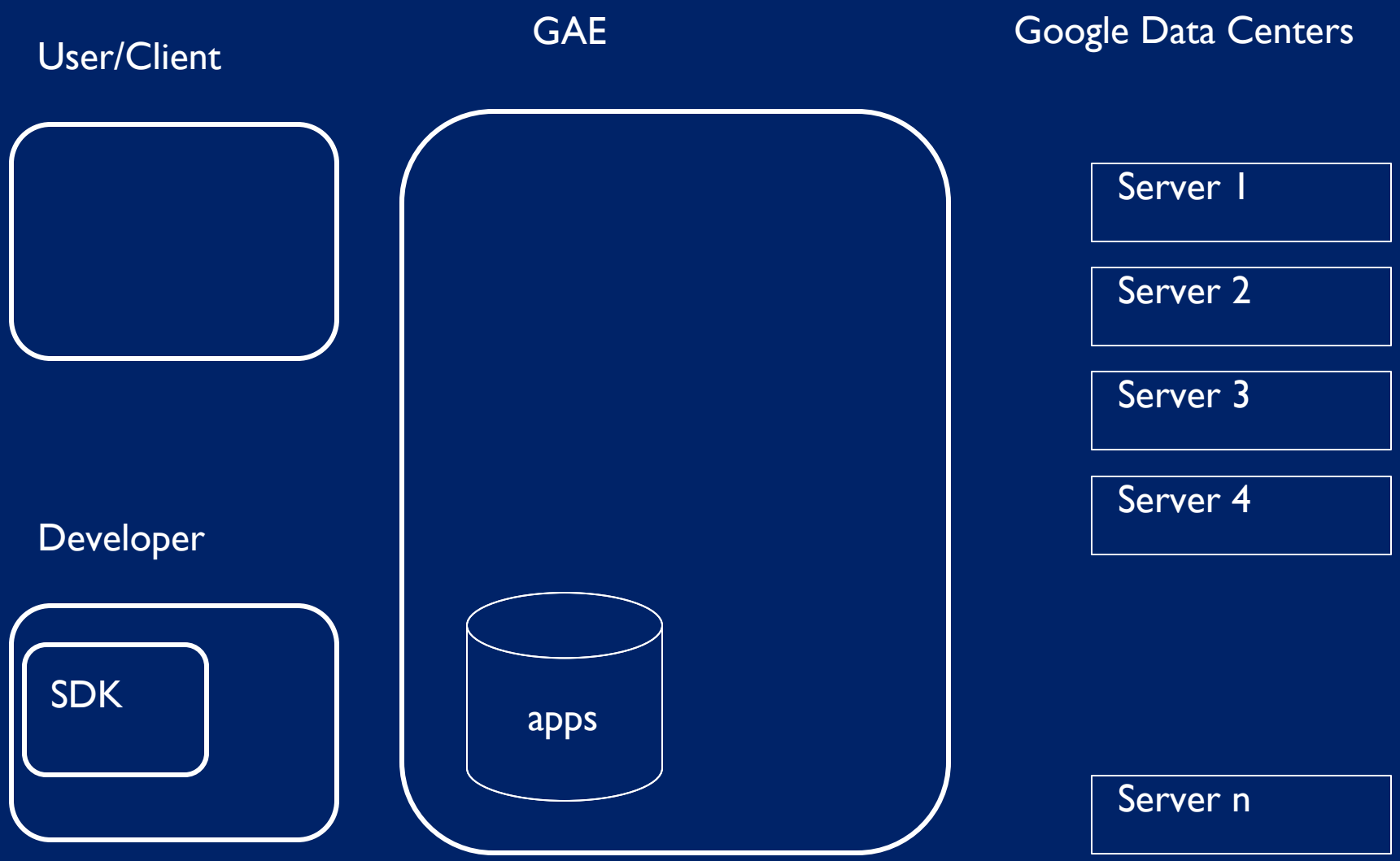
GAE Runtime Environment

Source: Sanderson (2013) Ch1.



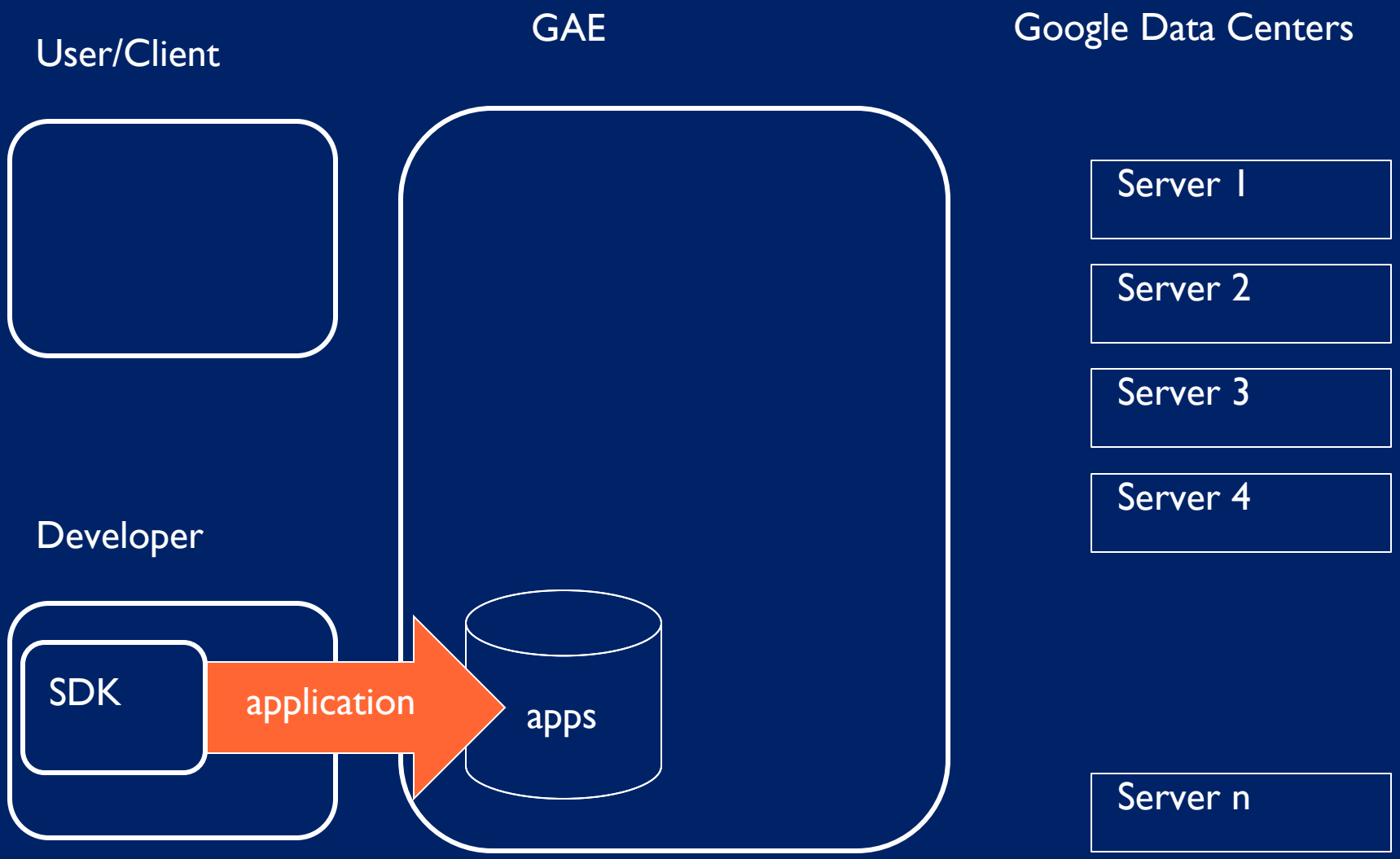
GAE Runtime Environment

Source: Sanderson (2013) Ch1.



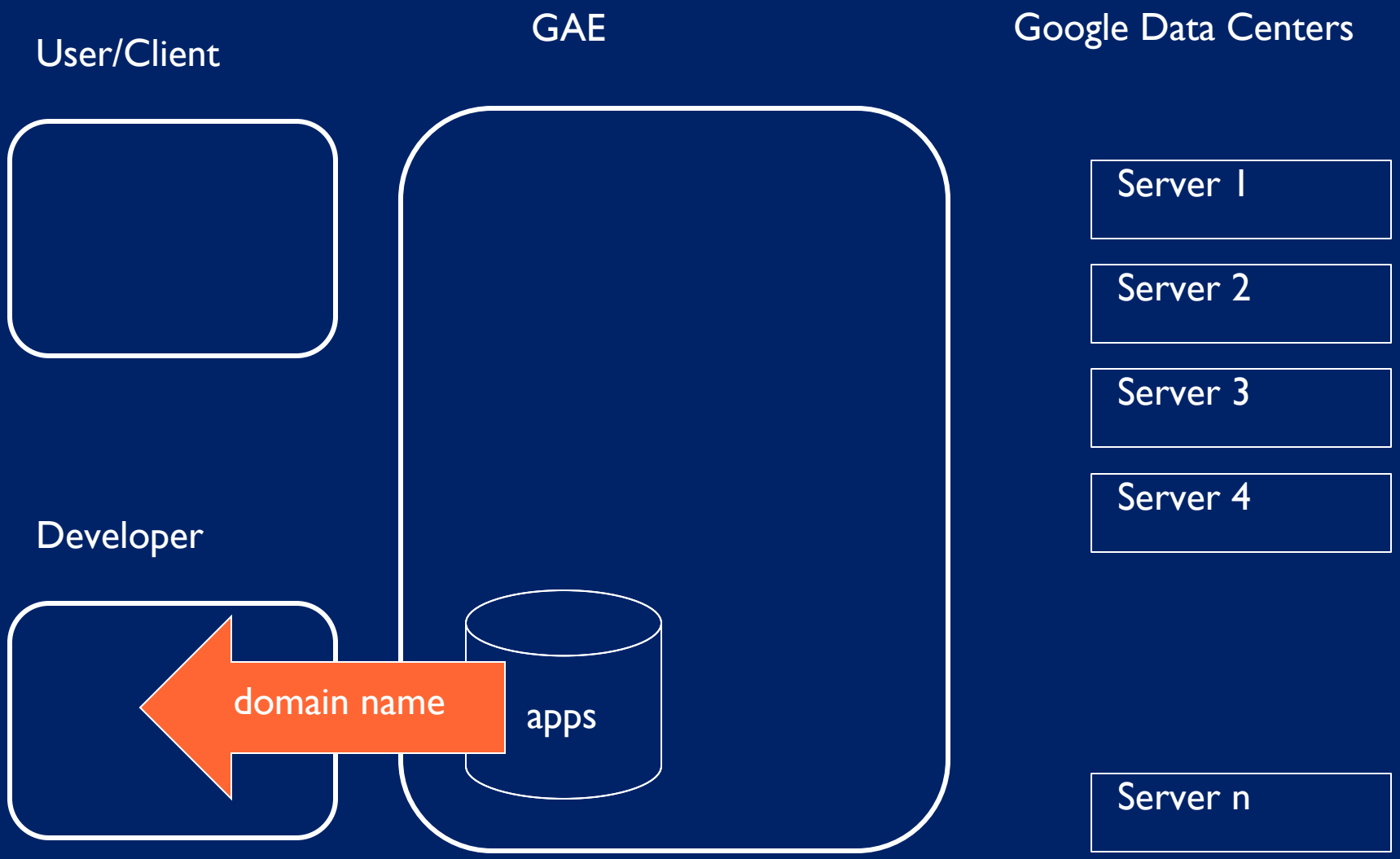
GAE Runtime Environment

Source: Sanderson (2013) Ch1.



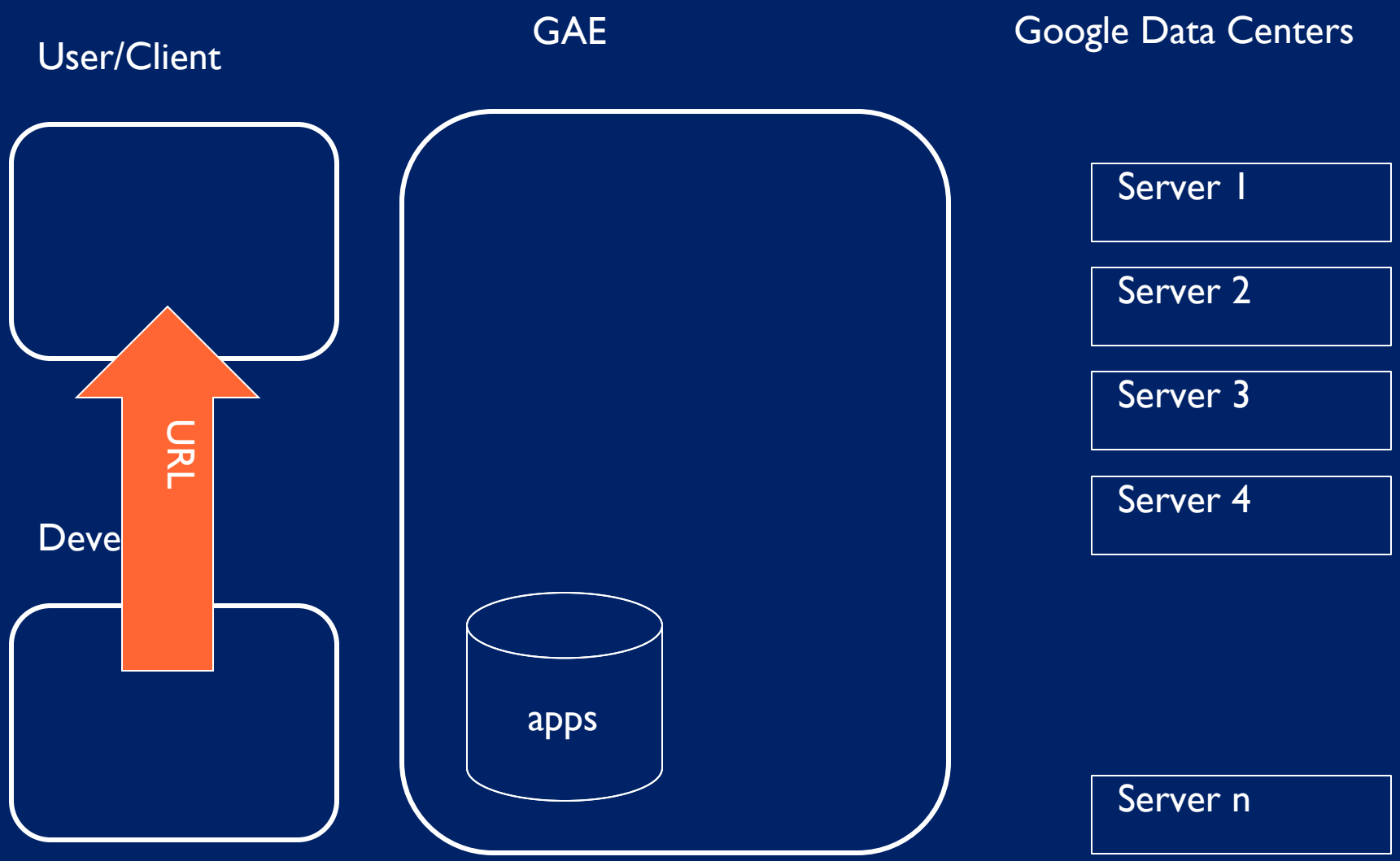
GAE Runtime Environment

Source: Sanderson (2013) Ch1.



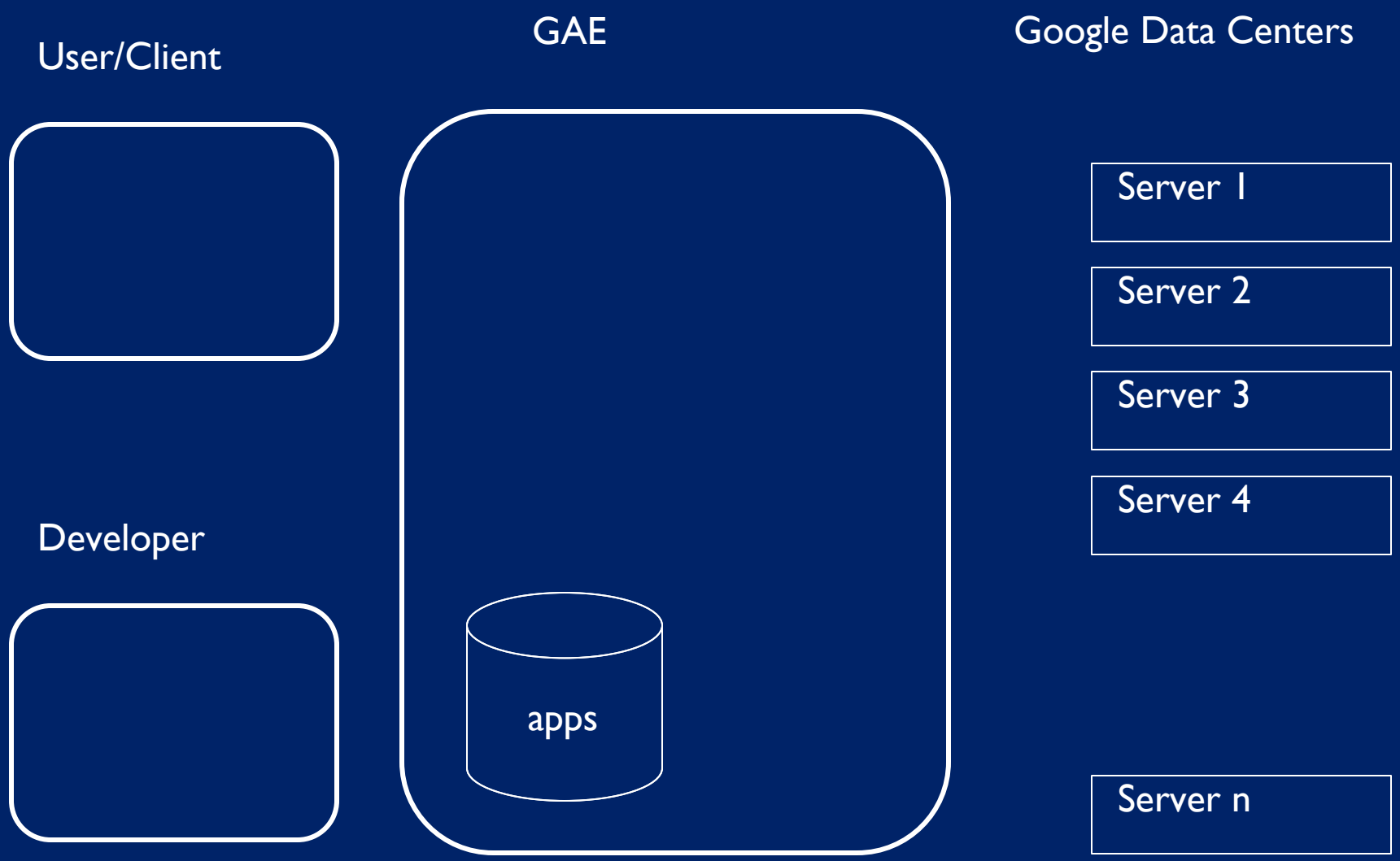
GAE Runtime Environment

Source: Sanderson (2013) Ch1.



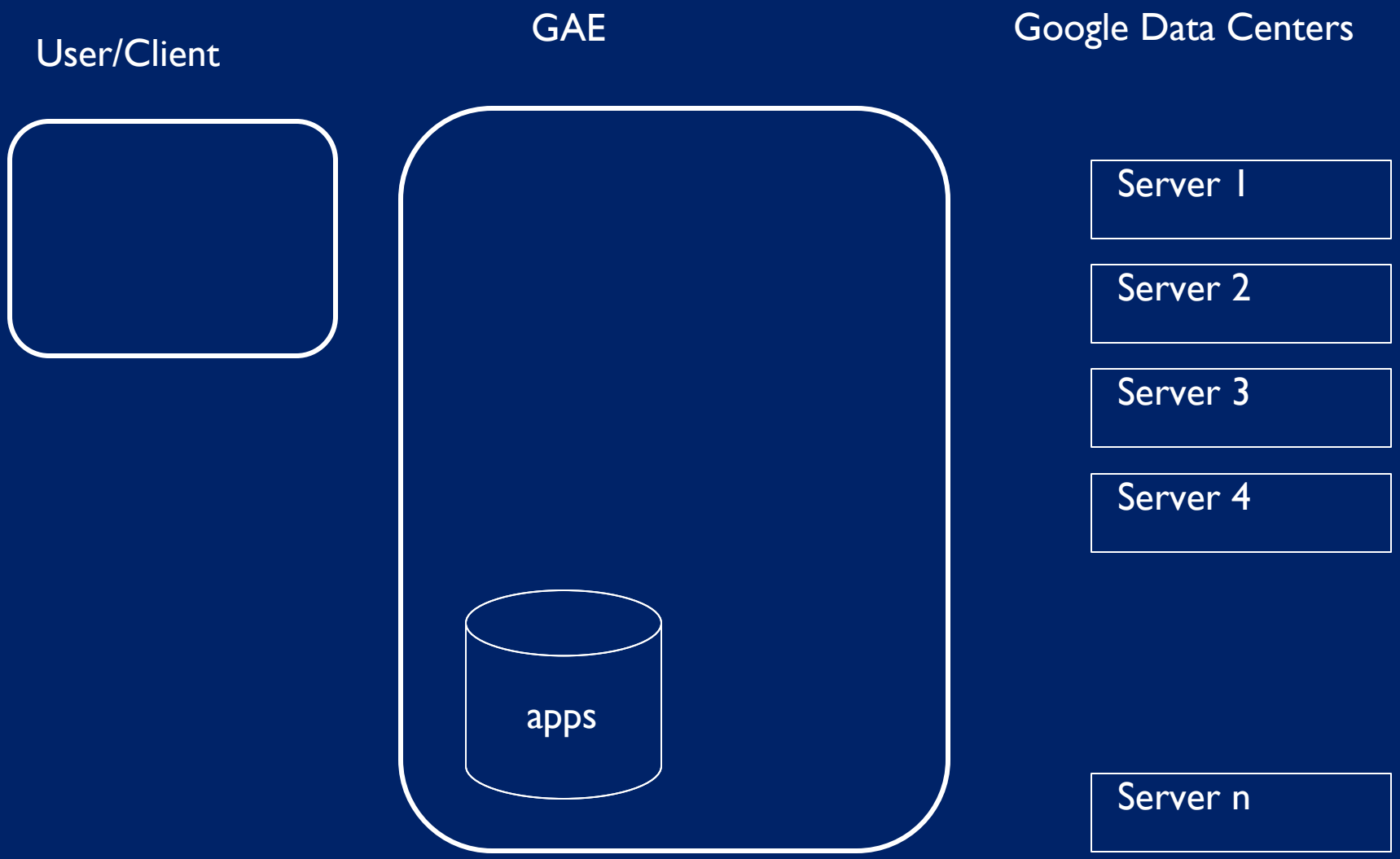
GAE Runtime Environment

Source: Sanderson (2013) Ch1.



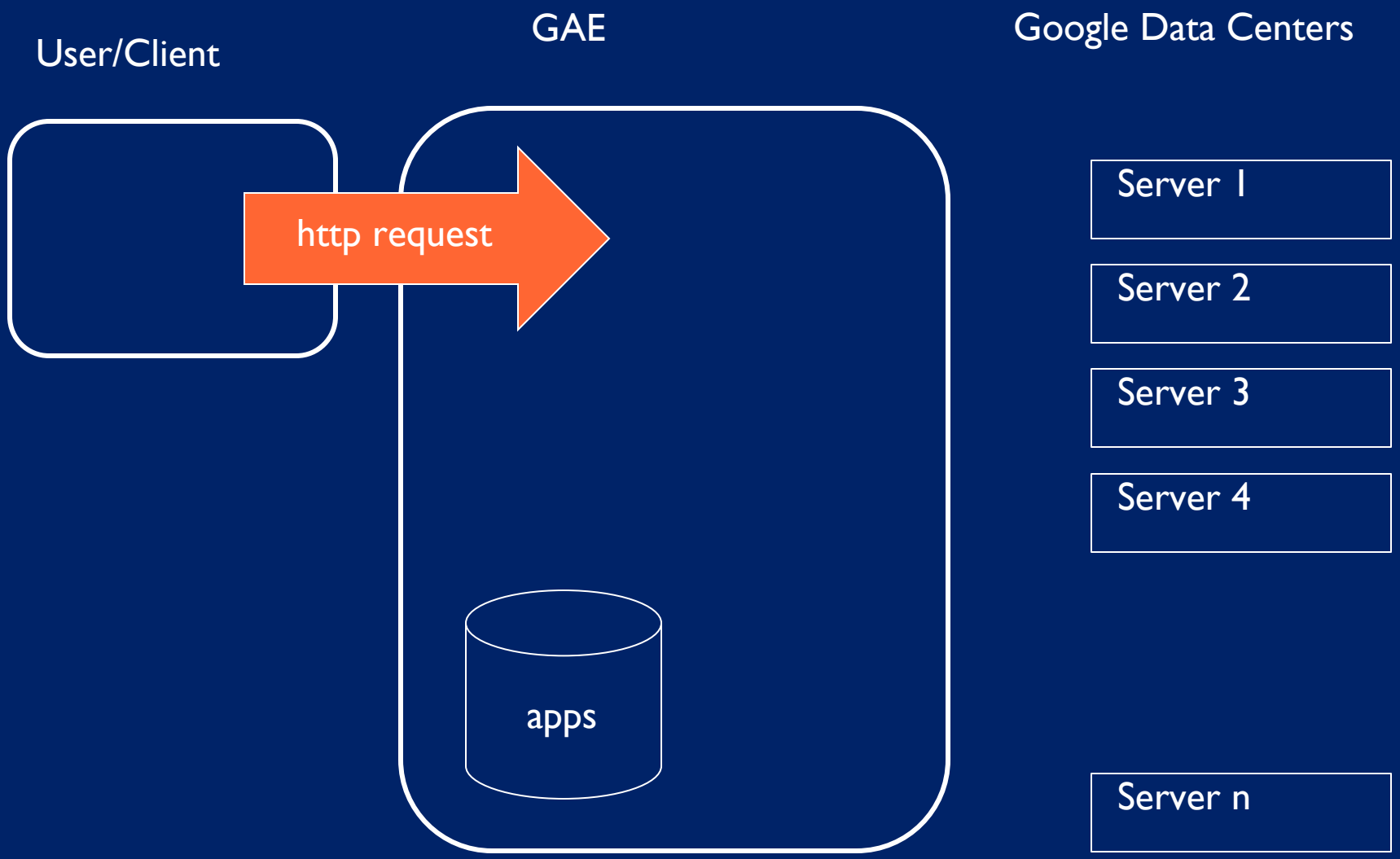
GAE Runtime Environment

Source: Sanderson (2013) Ch1.



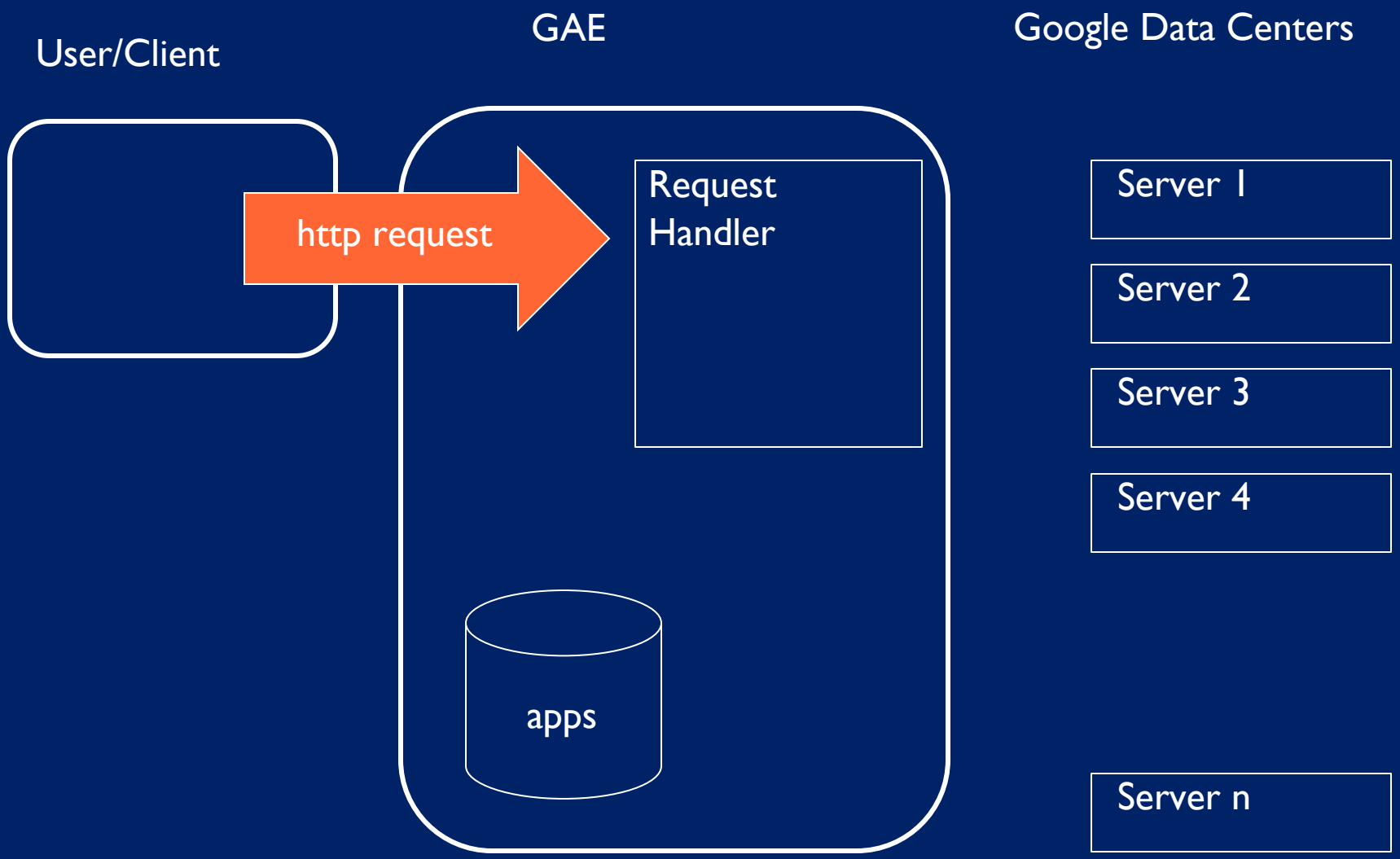
GAE Runtime Environment

Source: Sanderson (2013) Ch1.



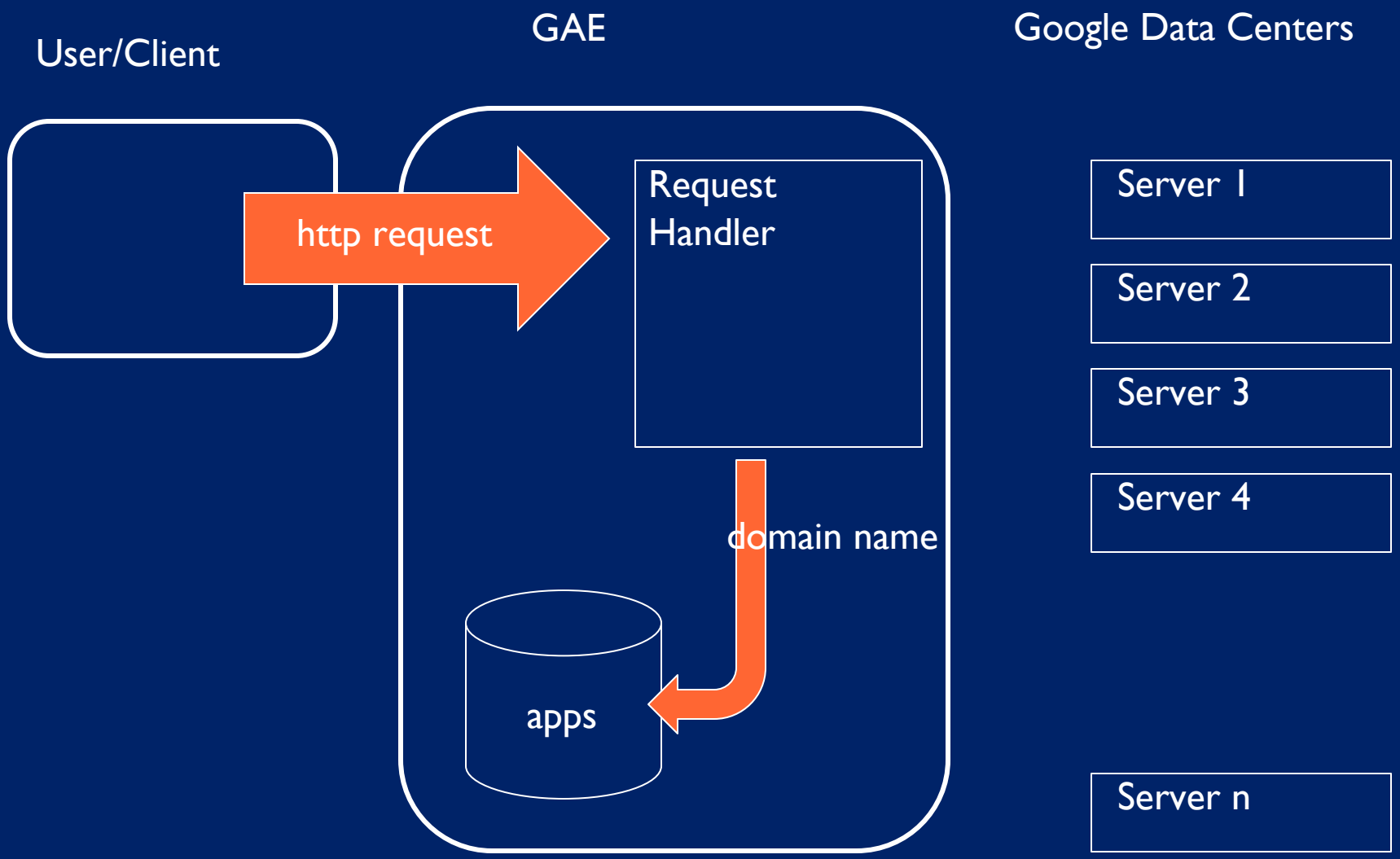
GAE Runtime Environment

Source: Sanderson (2013) Ch1.



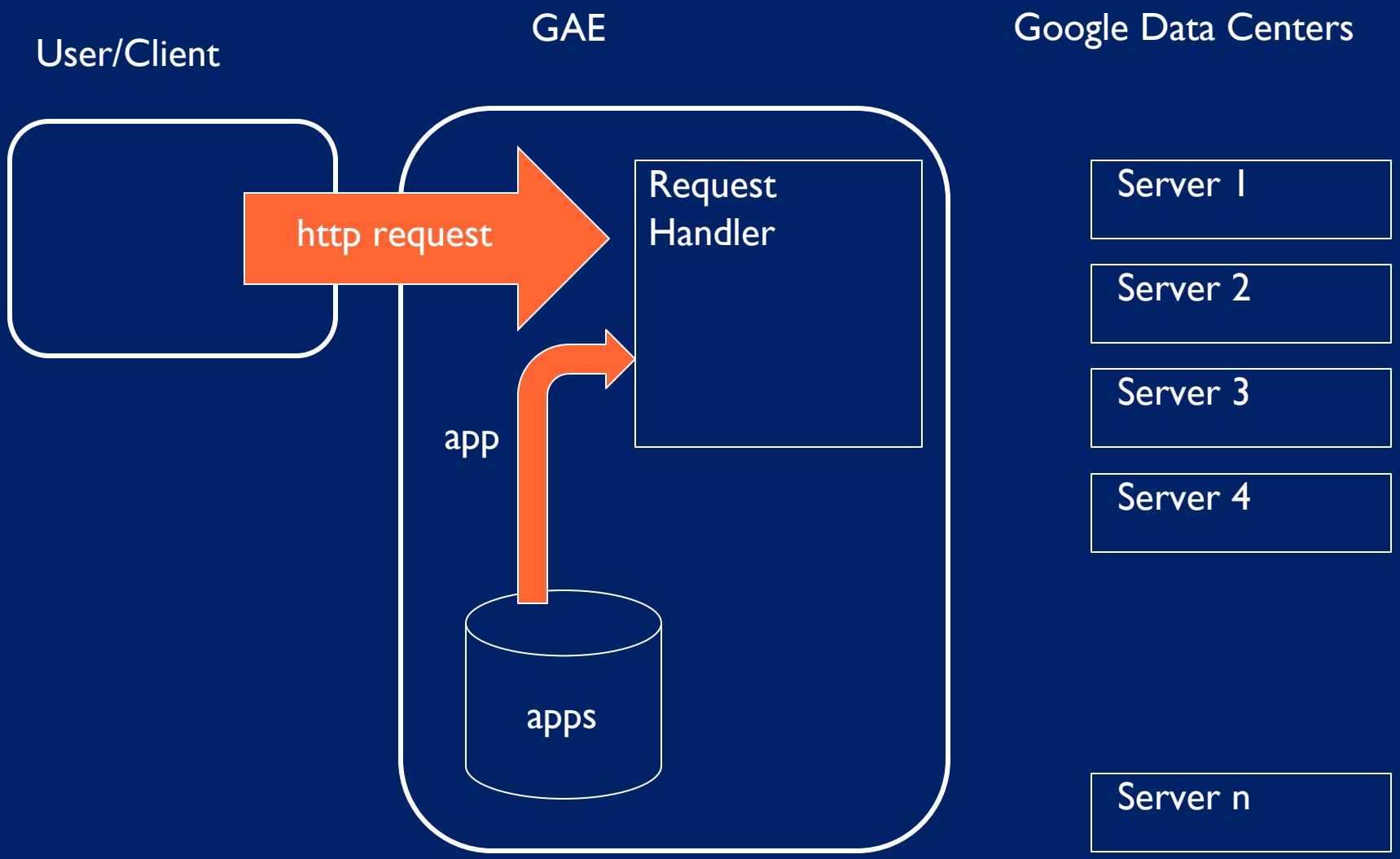
GAE Runtime Environment

Source: Sanderson (2013) Ch1.



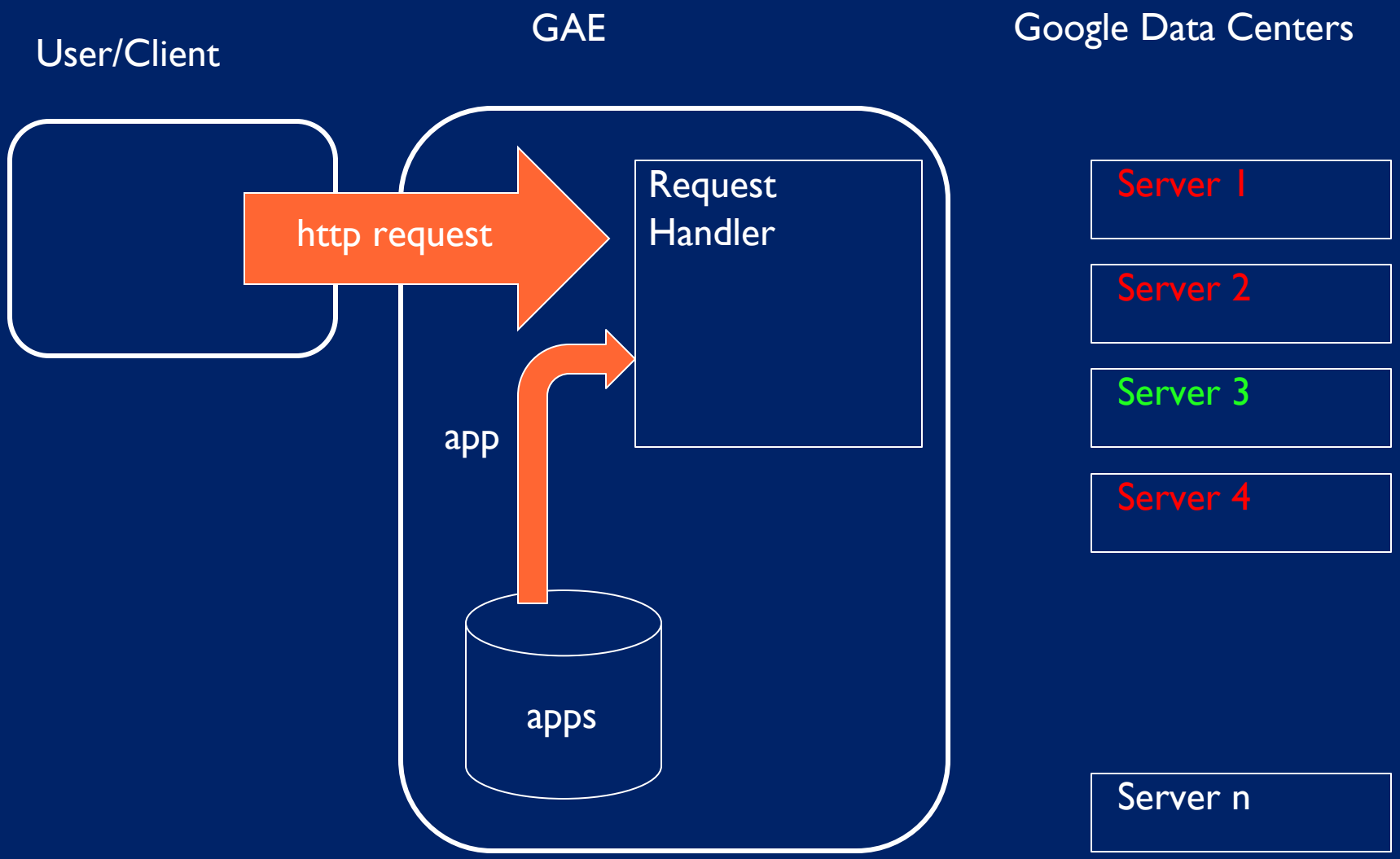
GAE Runtime Environment

Source: Sanderson (2013) Ch1.



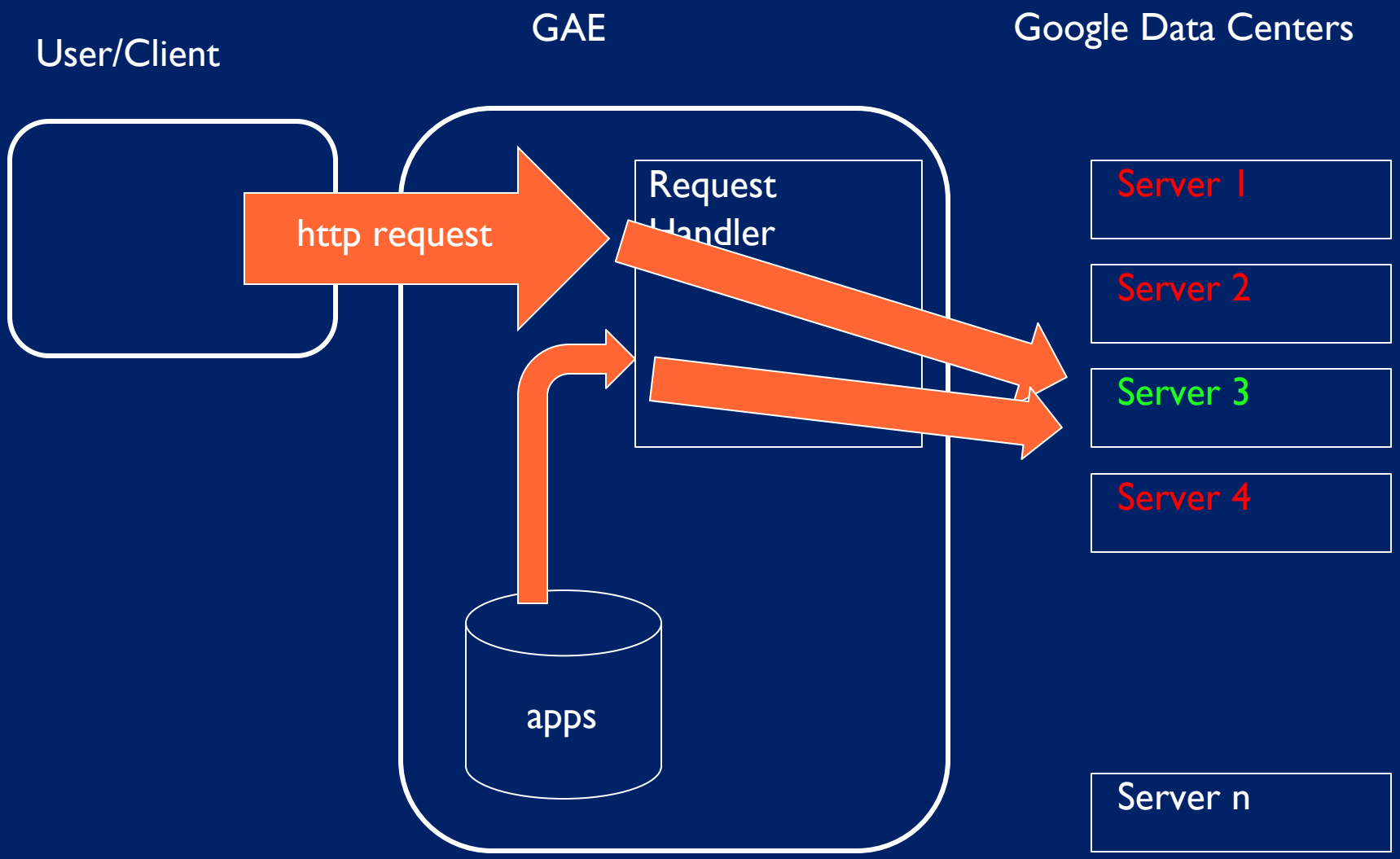
GAE Runtime Environment

Source: Sanderson (2013) Ch1.



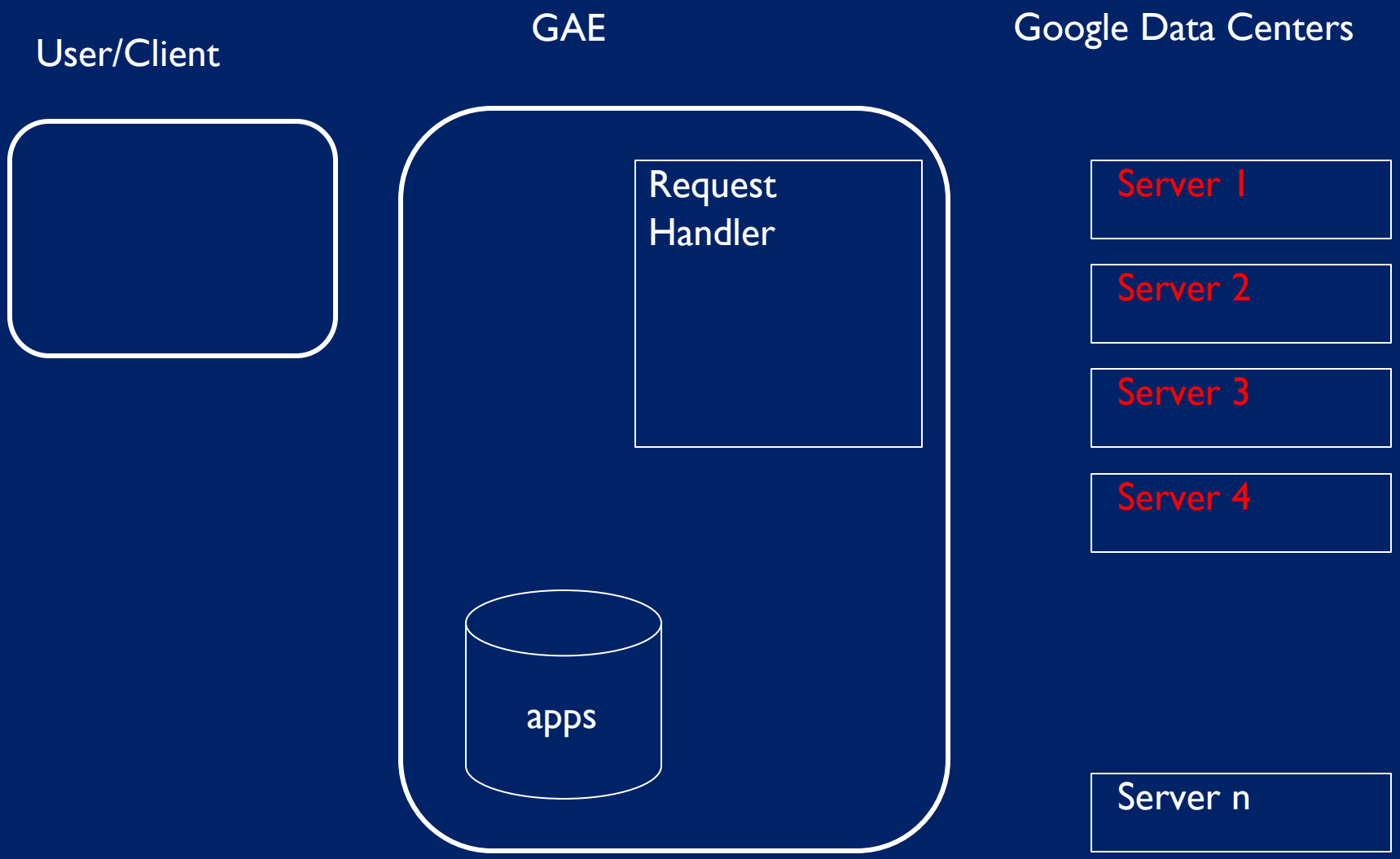
GAE Runtime Environment

Source: Sanderson (2013) Ch1.



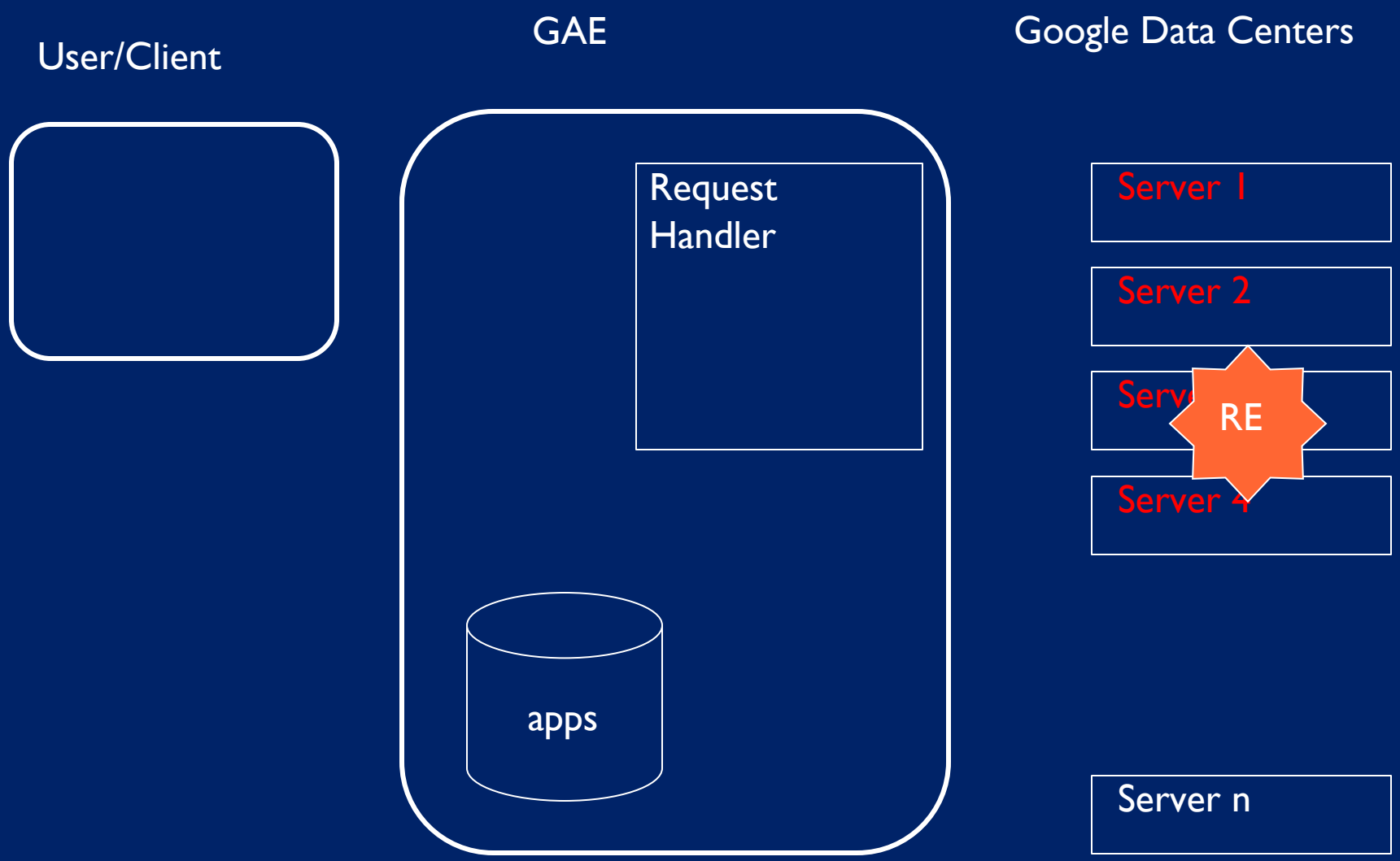
GAE Runtime Environment

Source: Sanderson (2013) Ch1.



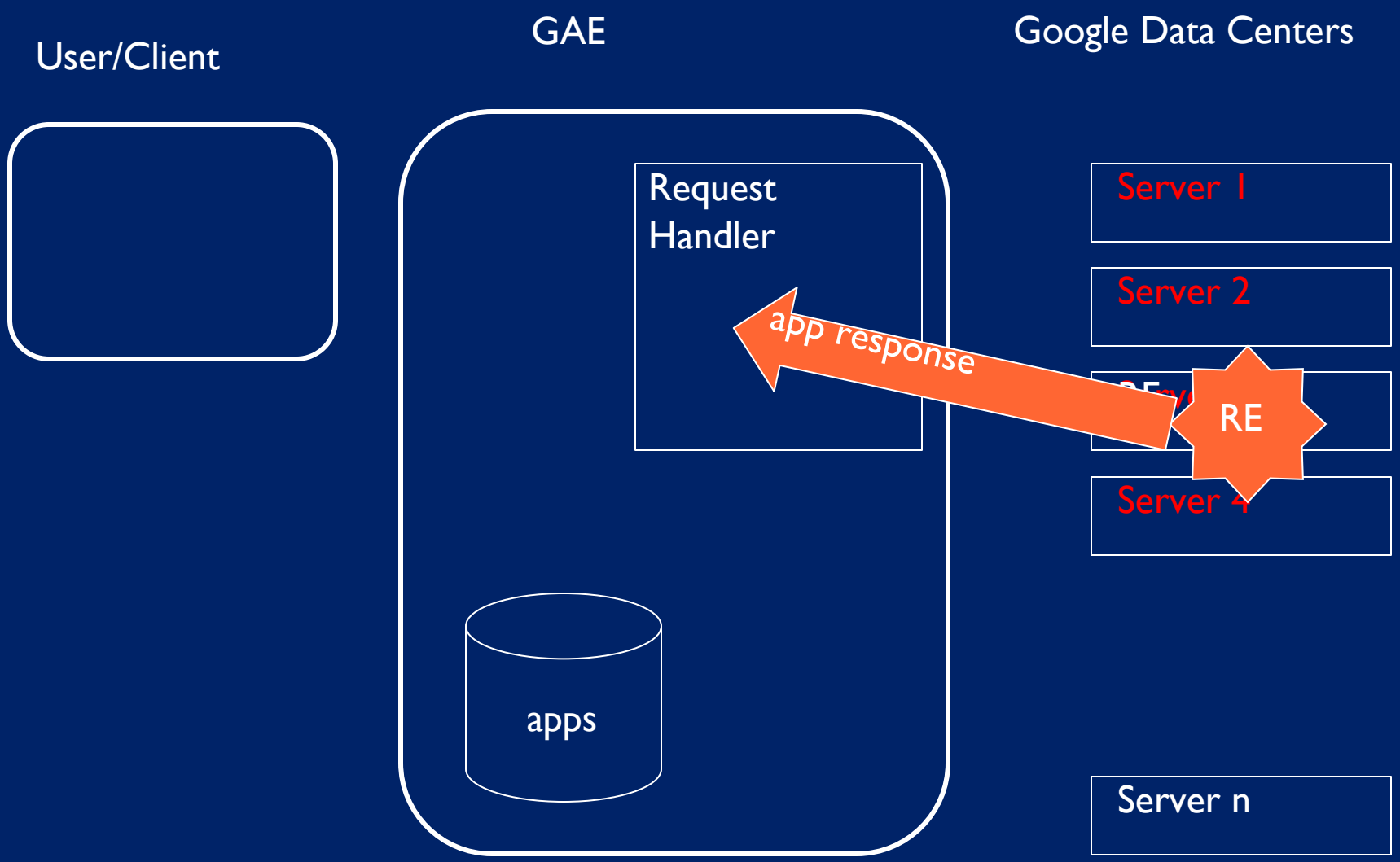
GAE Runtime Environment

Source: Sanderson (2013) Ch1.



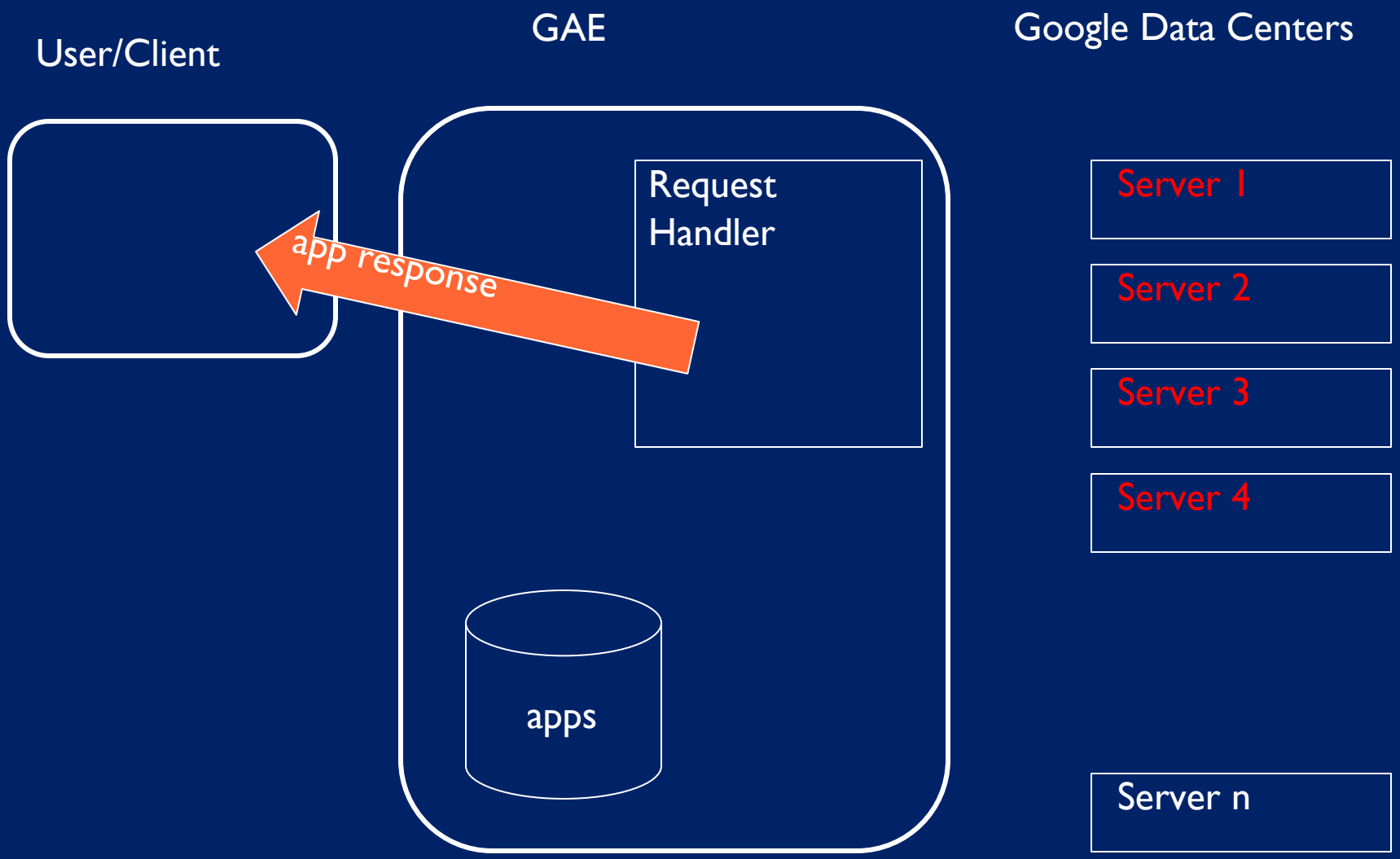
GAE Runtime Environment

Source: Sanderson (2013) Ch1.



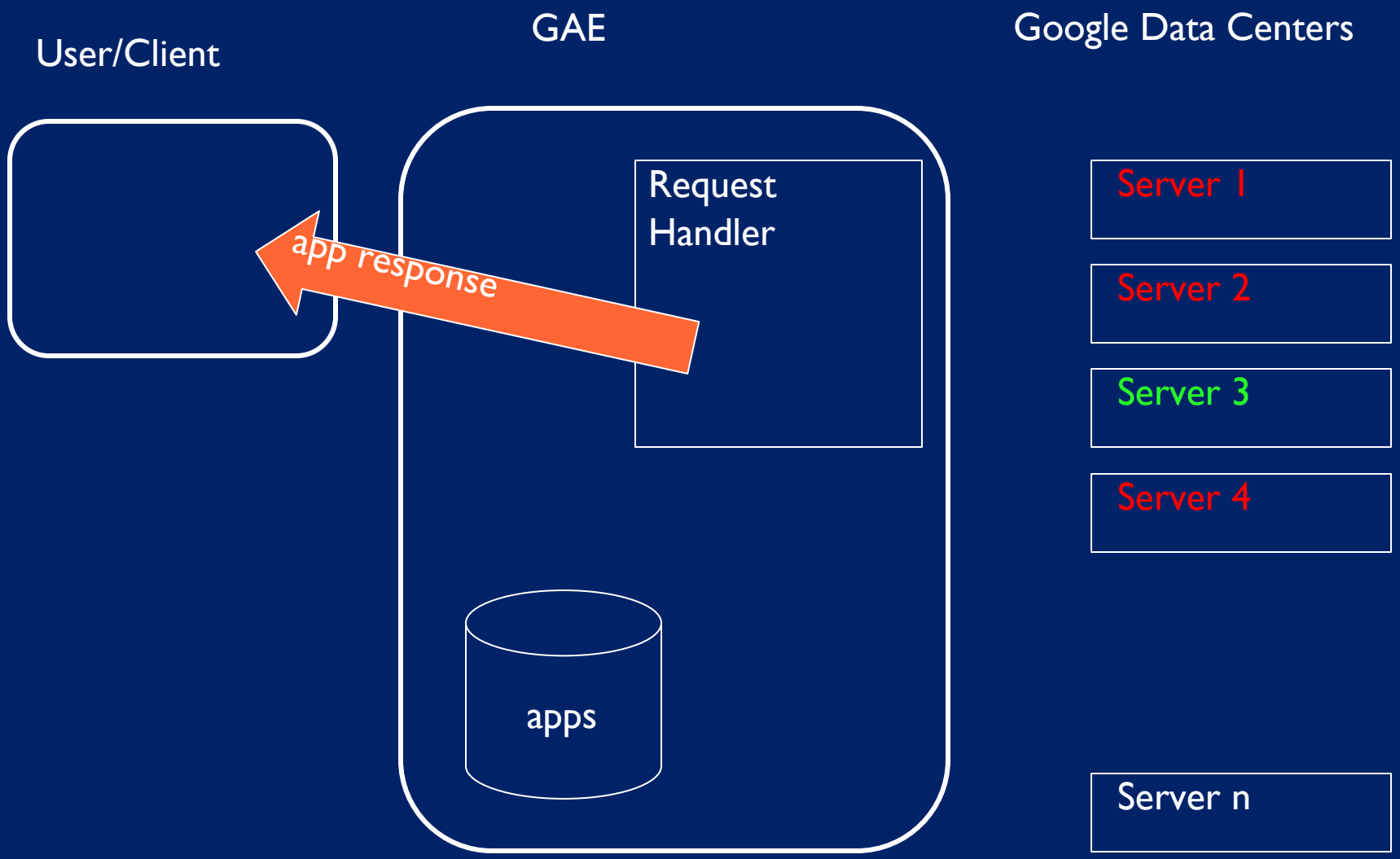
GAE Runtime Environment

Source: Sanderson (2013) Ch1.



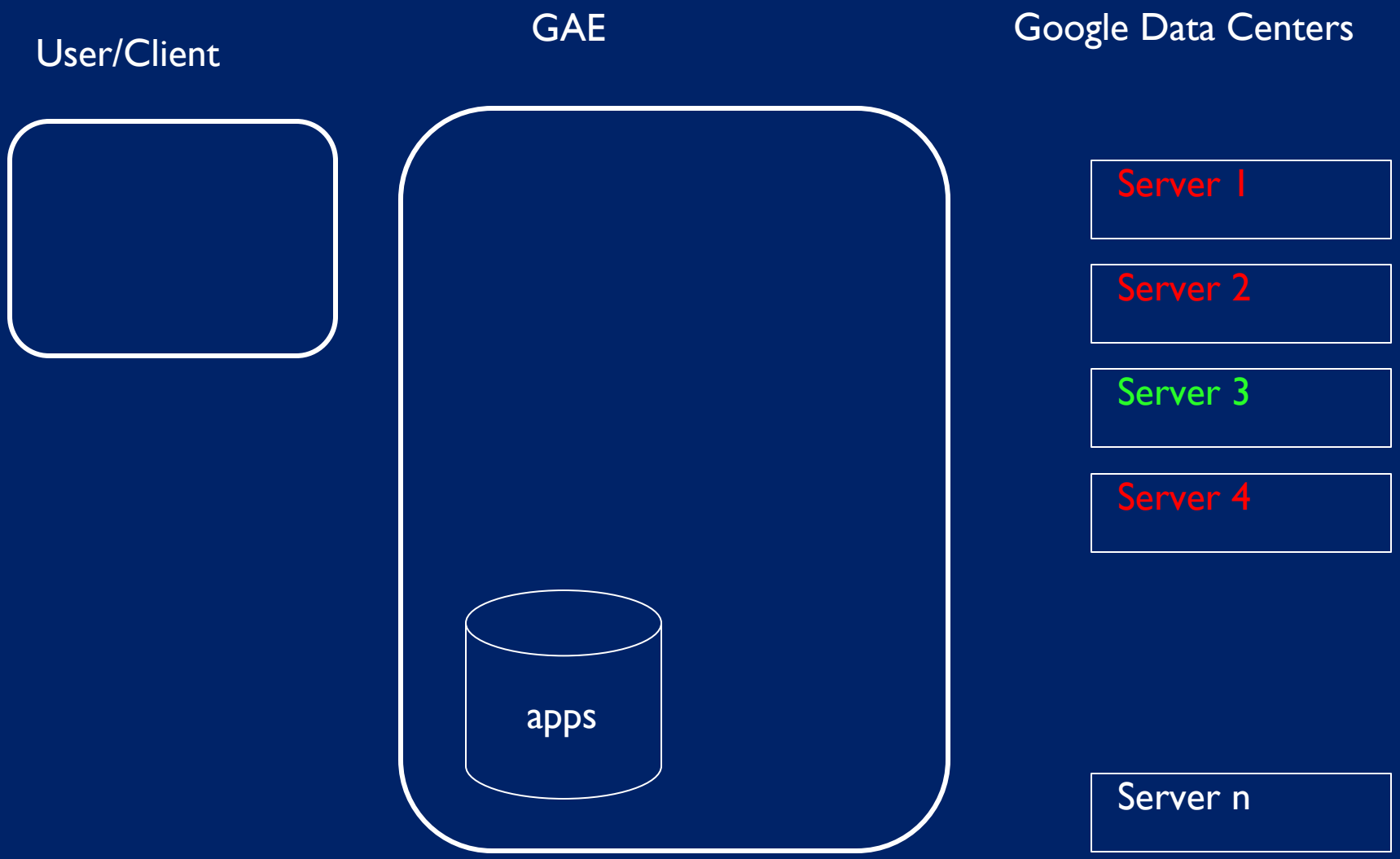
GAE Runtime Environment

Source: Sanderson (2013) Ch1.



GAE Runtime Environment

Source: Sanderson (2013) Ch1.



All this, usually in <1 second
(60 seconds max)

GAE Runtime Environment

The runtime environment (RE) only exists for as long as the request is being handled: RE disappears when the request-handler ends.

GAE provides methods for storing data between requests but they are outside the RE: the RE state is not retained between requests.

This allows GAE to distribute traffic over as many servers as it needs, regardless of how much traffic it is currently handling

- Assigns each app to server that is expected to give the fastest response

GAE automatically creates and destroys instances as needed to accommodate your app's traffic

Your app code cannot directly access the server on which it is running

- Can call services for reading/writing data, network access, etc
- Insulated from actions/crashes of other apps running on same server
- i.e., "sandboxed" – GAE RE also limits amount of memory & clock time…

GAE Runtime Environment

…Request Handler has to return a response to the client within 60 secs

But GAE is optimized to run applications that respond in <1 sec

GAE automated load-balancing means that if an app is CPU-heavy, that app may be slowed down (given fewer CPU cycles) to improve performance of other apps sharing that physical server… clock-time of your app may vary

How instances are created, initialized, and destroyed is part of the configuration for your app… the usual cost/performance trade-offs

- Requests can be queued to be processed by next available instance
- Or you can pay for more instances

You can also manually allocate & deallocate special "backend" or "server" instances, but no GAE-provided auto-scaling on these

GAE Static File Servers

Some web content (e.g. JavaScript code, style sheets, images, html files etc) is not expected to change in the regular operation of an app/site…

…known as static files

For efficiency, GAE provides separate servers for delivery of static files

You upload static files along with the app code

Several associated control parameters (e.g. cacheing options)

GAE Datastore

Storing data, holding it from one request to another, is clearly useful.

Pre-cloud: a single database server would be scaled up (and up, and up…)

- …Almost always a relational database

GAE approach, the Datastore, is cloud style: scale-out, not scale-up

Datastore scale-out is automated – you don't have to worry about how it is distributed over multiple servers

But you have to sacrifice relational database Join-Query mode of working

GAE Datastore holds data as entities;

Each entity has one or more properties,

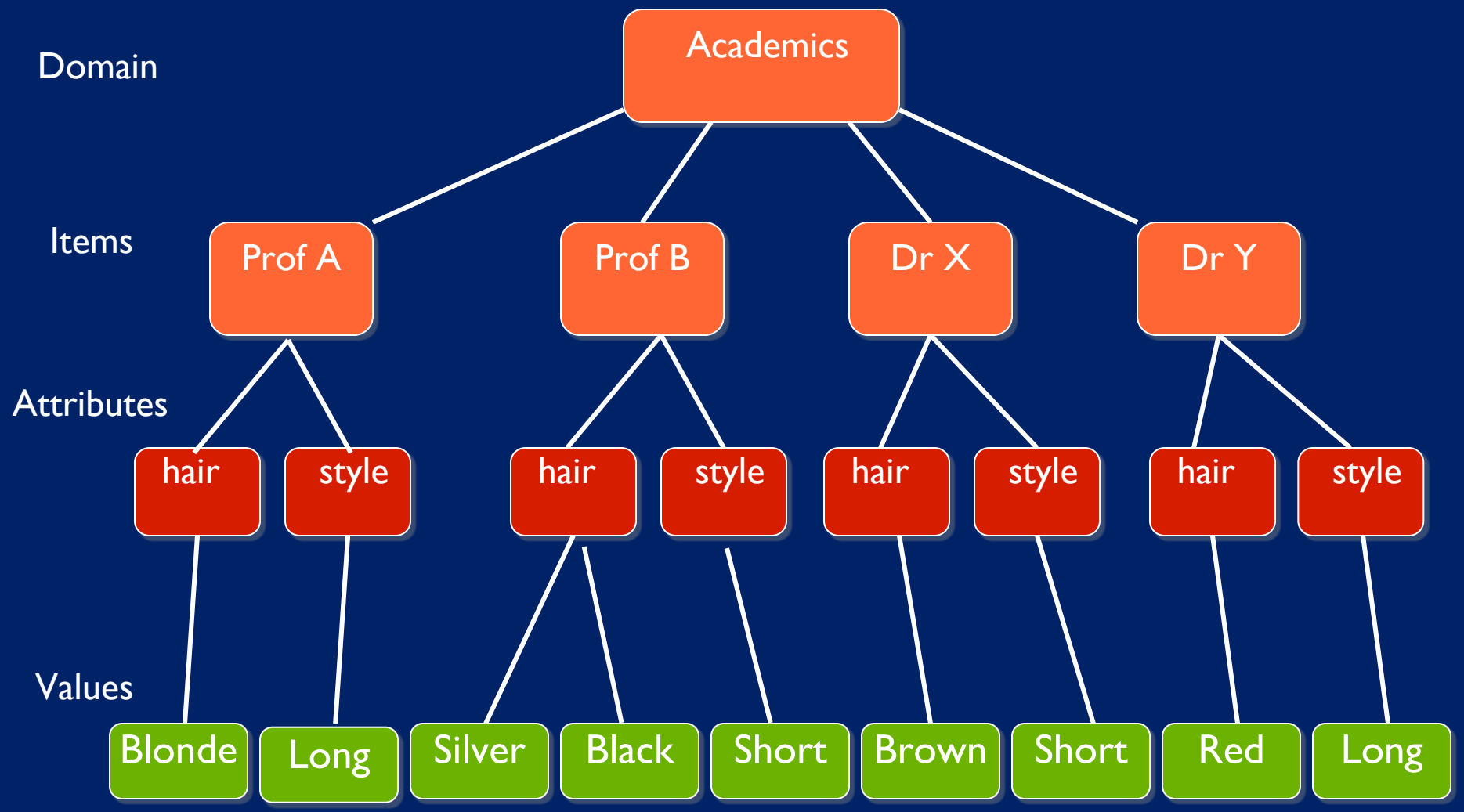
Each property has a name, and one or more values (the value is one of several primitive types);

Entity belongs to a kind (like a category or class) + a unique key (within kind)

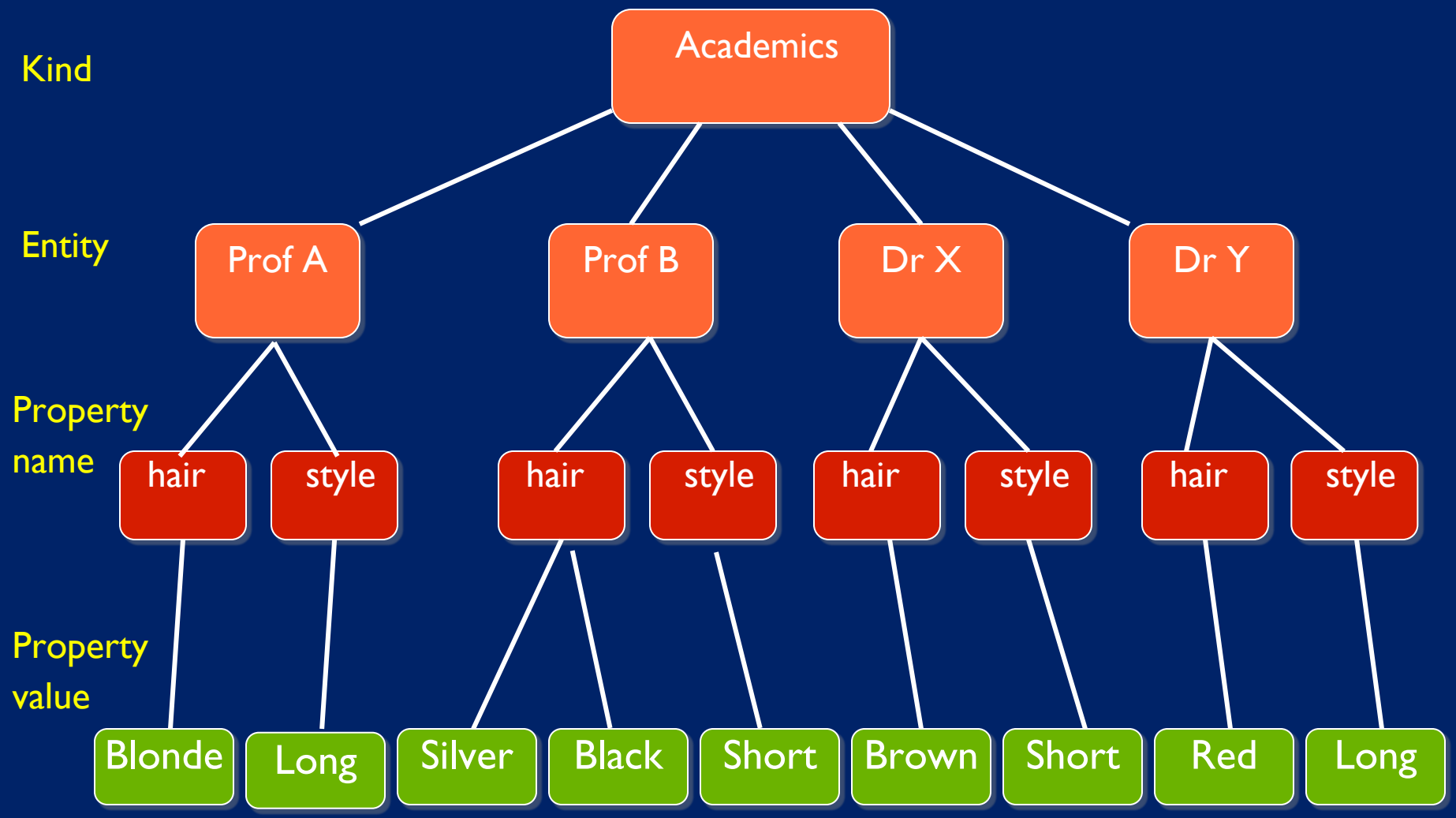
GAE Datastore

Remember AWS SimpleDB from start of this lecture…

Amazon SimpleDB: an example hierarchy



GAE Datastore



GAE Datastore: Queries and (pre-)Indexes

Source: Sanderson (2013) Ch1.

Datastore queries return entities of a single kind, or just their keys

GQL queries can filter entities on the basis of their property values

GQL query results can be returned in sorted order, of property-values

GQL can filter/sort on keys too

Big idea: query pre-indexing

For every allowable query, the datastore pre-builds an index

- GAE SDK watches which queries are used in development & testing
- When you upload your app, the datastore makes indexes for every query that the SDK saw during developer's testing (this list can be edited)
- Means resolving/answering queries is very fast
- But adding/updating entities can create a lot of book-keeping

GAE Datastore

Remember this slide from Lecture 1…

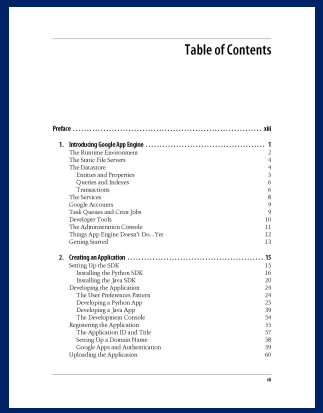
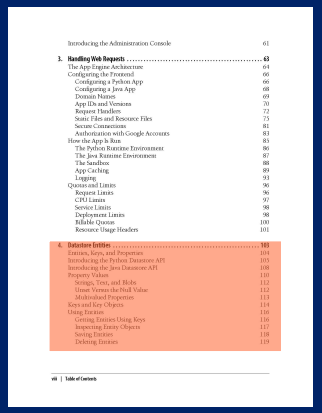


Table of Contents

Preface xiii

1. Introducing Google App Engine 1
The Runtime Environment 2
The Static File Servers 4
The Datastore 4
Entities and Properties 5
Queries and Indexes 6
Transactions 6
The Services 8
Google Accounts 9
Task Queues and Cron Jobs 9
Developer Tools 10
The Administration Console 11
Things App Engine Doesn't Do...Yet 12
Getting Started 13

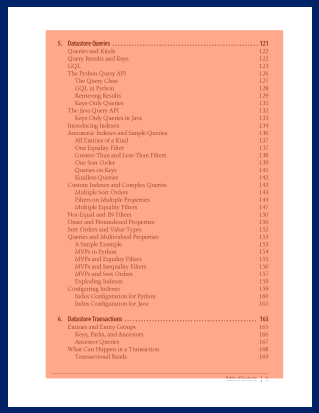
2. Creating an Application 15
Setting Up the SDK 15
Installing the Python SDK 16
Installing the Java SDK 20
Developing the Application 24
The User Preferences Pattern 24
Developing a Python App 25
Developing a Java App 39
The Development Console 54
Registering the Application 55
The Application ID and Title 57
Setting Up a Domain Name 58
Google Apps and Authentication 59
Uploading the Application 60



Introducing the Administration Console 61

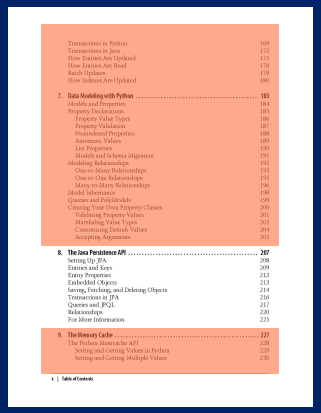
3. Handling Web Requests 63
The App Engine Architecture 64
Configuring the Frontend 66
Configuring a Python App 66
Configuring a Java App 68
Domain Names 69
App IDs and Versions 70
Request Handlers 72
Static Files and Resource Files 75
Secure Connections 81
Authorization with Google Accounts 83
How the App Is Run 83
The Python Runtime Environment 86
The Java Runtime Environment 87
The Sandbox 88
App Caching 89
Logging 93
Quotas and Limits 96
Request Limits 96
CPU Limits 97
Service Limits 98
Deployment Limits 98
Billable Quotas 100
Resource Usage Headers 101

4. Datastore Entities 103
Entities, Keys, and Properties 104
Introducing the Python Datastore API 105
Introducing the Java Datastore API 108
Property Values 110
Strings, Text, and Blobs 112
Unset Versus the Null Value 112
Multivalued Properties 113
Keys and Key Objects 114
Using Entities 116
Getting Entities Using Keys 116
Inspecting Entity Objects 117
Saving Entities 118
Deleting Entities 119



5. Datastore Queries 121
Queries and Kinds 122
Query Results and Keys 122
GQL 123
The Python Query API 126
The Query Class 127
GQL in Python 128
Retrieving Results 129
Keys-Only Queries 131
The Java Query API 132
Keys-Only Queries in Java 133
Introducing Indexes 134
Automatic Indexes and Simple Queries 136
All Entities of a Kind 137
One Equality Filter 137
Greater-Than and Less-Than Filters 138
One Sort Order 139
Queries on Keys 141
Kindless Queries 142
Custom Indexes and Complex Queries 143
Multiple Sort Orders 143
Filters on Multiple Properties 144
Multiple Equality Filters 147
Not-Equal and IN Filters 150
Unset and Nonindexed Properties 150
Sort Orders and Value Types 152
Queries and Multivalued Properties 153
A Simple Example 153
AND's in Python 154
AND's and Equality Filters 155
AND's and Inequality Filters 156
AND's and Sort Orders 157
Exploding Indexes 159
Configuring Indexes 159
Index Configuration for Python 160
Index Configuration for Java 161

6. Datastore Transactions 163
Entities and Entity Groups 165
Keys, Paths, and Ancestors 166
Ancestor Queries 167
What Can Happen in a Transaction 168
Transactional Reads 169



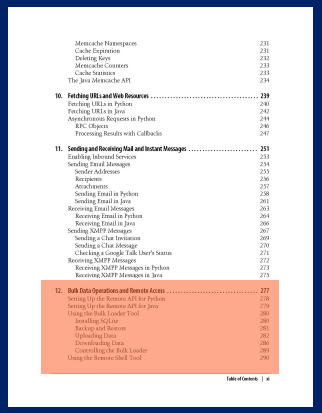
Transactions in Python 169
Transactions in Java 172
How Entities Are Updated 175
How Entities Are Read 178
Batch Updates 179
How Indexes Are Updated 180

7. Data Modeling with Python 183
Models and Properties 184
Property Declarations 185
Property Value Types 186
Property Validation 187
Nonindexed Properties 188
Automatic Values 189
List Properties 190
Models and Schema Migration 191
Modeling Relationships 192
One-to-Many Relationships 193
One-to-One Relationships 195
Many-to-Many Relationships 196
Model Inheritance 198
Queries and Polymodels 199
Creating Your Own Property Classes 200
Validating Property Values 201
Marshaling Value Types 202
Customizing Default Values 203
Accepting Arguments 203

8. The Java Persistence API 207
Setting Up JPA 208
Entities and Keys 209
Entity Properties 212
Embedded Objects 213
Saving, Fetching, and Deleting Objects 214
Transactions in JPA 216
Queries and JPQL 217
Relationships 220
For More Information 223

9. The Memory Cache 227
The Python Memcache API 228
Setting and Getting Values in Python 229
Setting and Getting Multiple Values 230

x | Table of Contents



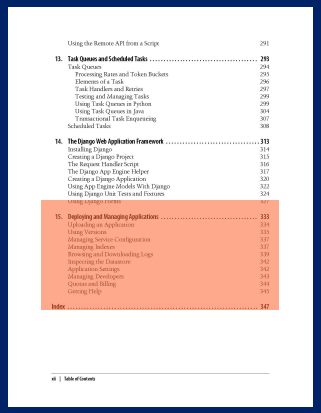
Memcache Namespaces 231
Cache Expiration 231
Deleting Keys 232
Memcache Counters 233
Cache Statistics 233
The Java Memcache API 234

10. Fetching URLs and Web Resources 239
Fetching URLs in Python 240
Fetching URLs in Java 242
Asynchronous Requests in Python 244
RPC Objects 246
Processing Results with Callbacks 247

11. Sending and Receiving Mail and Instant Messages 251
Enabling Inbound Services 253
Sending Email Messages 254
Sender Addresses 255
Recipients 256
Attachments 257
Sending Email in Python 258
Sending Email in Java 261
Receiving Email Messages 263
Receiving Email in Python 264
Receiving Email in Java 266
Sending XMPP Messages 267
Sending a Chat Invitation 269
Sending a Chat Message 270
Checking a Google Talk User's Status 271
Receiving XMPP Messages 272
Receiving XMPP Messages in Python 273
Receiving XMPP Messages in Java 275

12. Bulk Data Operations and Remote Access 277
Setting Up the Remote API for Python 278
Setting Up the Remote API for Java 279
Using the Bulk Loader Tool 280
Installing SQLite 280
Backup and Restore 281
Uploading Data 282
Downloading Data 286
Controlling the Bulk Loader 289
Using the Remote Shell Tool 290

Table of Contents | xi



Using the Remote API from a Script 291

13. Task Queues and Scheduled Tasks 293
Task Queues 294
Processing Rates and Token Buckets 295
Elements of a Task 296
Task Handlers and Retries 297
Testing and Managing Tasks 299
Using Task Queues in Python 299
Using Task Queues in Java 304
Transactional Task Enqueueing 307
Scheduled Tasks 308

14. The Django Web Application Framework 313
Installing Django 314
Creating a Django Project 315
The Request Handler Script 316
The Django App Engine Helper 317
Creating a Django Application 320
Using App Engine Models With Django 322
Using Django Unit Tests and Fixtures 324
Using Forms 326

15. Deploying and Managing Applications 333
Uploading an Application 334
Using Versions 335
Managing Service Configuration 337
Managing Indexes 337
Browsing and Downloading Logs 339
Inspecting the Datastore 342
Application Settings 342
Managing Developers 343
Quotas and Billing 344
Getting Help 345

Index 347

xii | Table of Contents

>40% of this book is all about "dataSTORES"

GAE Datastore

(As you can imagine, we are skipping over some details here)

GAE Datastore

(As you can imagine, we are skipping over some details here)

App instances in the RE access the datastore as a service, via an API.

The datastore self-scales, but the app doesn't need to know how.

Other self-scaling GAE services

Memcache

- Short-term key-value storage service
- Uses RAM not disk; hence much faster than datastore
- Unlike datastore, it is not persistent… server down => data-loss
- Best used as a cache – hold recent results from frequent actions

BLOBstore

- Storage for large items, like images, video, or other big files

URL Fetch

- HTTP request to another server on the Internet

Mail & Instant Messaging

- IM to/from chat services that use XMPP protocol, like Google Talk

Channels

- Real-time two-way comms direct with web-browser

And on and on and on

Sanderson has a 1h19m introductory webcast for GAE + Python:

http://oreillynet.com/pub/e/2212

Chapter 2 of Sanderson (2012) is a walk-through of creating a GAE app.

And lots of training/tutorial videos at:

https://www.youtube.com/user/googlecloudplatform

Shedloads of sample code at:

http://googlecloudplatform.github.io/

Video Tutorial/Demo

9m29s tutorial video from GAE (this guy types very fast…)

http://www.youtube.com/watch?v=bfgO-LXGpTM

NB1: YAML is a human-readable data-file format

NB2: This is from April 2008, some details may differ in current release

And finally…

Another slide from Lecture 1…

A common (and useful) stratification: *aaS

- SaaS: Software as a Service
- End-user application software functionality remotely delivered over the internet/web
- You used to buy this as a CD/DVD in a shrink-wrapped cardboard box

- PaaS: Platform as a Service
- Developer application software (middleware) functionality, remotely accessible
- E.g. provide a particular combination of OS, web-server, data-base, & scripting
- Popular instance is the free LAMP stack – Linux, Apache, MySQL, and PHP/Perl/Python)

GOOGLE APP ENGINE

- IaaS: Infrastructure as a Service
- IT infrastructure, almost always virtualized, remotely accessible, "bare metal"

AMAZON EC2, S3, SQS

GOOGLE COMPUTE ENGINE

Google Cloud Platform

Go to my console | Sign out

Search this site

Why Google | Products | Solutions | Customers | Documentation | Support | Partners | Blog

Try it now | Contact Sales

Compute Engine

Run large-scale workloads on virtual machines hosted on Google's infrastructure. Choose a VM that fits your needs and gain the performance of Google's worldwide fiber network.

Get Started

Features | Case Studies | Pricing Calculator | Pricing | Documentation

Features

High-performance virtual machines

Compute Engine's Linux VMs are consistently performant, scalable, highly secure and reliable. Supported distros include Debian and CentOS. You can choose from micro-VMs to large instances.

Powered by Google's global network

Create large compute clusters that benefit from strong and consistent cross-machine bandwidth. Connect to machines in other data centers and to other Google services using Google's private global fiber network.

(Really) Pay for what you use

Google bills in minute-level increments (with a 10-minute minimum charge), so you don't pay for unused computing time. With sustained use discounts, we automatically give you discounted prices for long-running

2015

Pricing

All machine types are charged a minimum of **10 minutes.** For example, if you run your instance for 2 minutes, you will be billed for 10 minutes of usage. After 10 minutes, instances are charged in **1 minute increments, rounded up to the nearest minute.** For example, an instance that lives for 11.25 minutes will be charged for 12 minutes of usage.

If you would like to discuss pricing for long-term commitments, please contact sales.

Machine Type Pricing

Standard

Instance type	Virtual Cores	Memory	Price (US$)/Hour (US hosted)	Price (US$)/Hour (Europe/APAC hosted)
n1-standard-1	1	3.75GB	$0.063	$0.069
n1-standard-2	2	7.5GB	$0.126	$0.138
n1-standard-4	4	15GB	$0.252	$0.276
n1-standard-8	8	30GB	$0.504	$0.552
n1-standard-16	16	60GB	$1.008	$1.104

High Memory

Machines for tasks that require more memory relative to virtual cores.

Instance type	Virtual Cores	Memory	Price (US$)/Hour	Price (US$)/Hour

2015

High Memory

Machines for tasks that require more memory relative to virtual cores.

Instance type	Virtual Cores	Memory	Price (US$)/Hour (US hosted)	Price (US$)/Hour (Europe/APAC hosted)
n1-highmem-2	2	13GB	$0.148	$0.162
n1-highmem-4	4	26GB	$0.296	$0.324
n1-highmem-8	8	52GB	$0.592	$0.648
n1-highmem-16	16	104GB	$1.184	$1.296

High CPU

Machines for tasks that require more virtual cores relative to memory.

Instance type	Virtual Cores	Memory	Price (US$)/Hour (US hosted)	Price (US$)/Hour (Europe/APAC hosted)
n1-highcpu-2	2	1.80GB	$0.080	$0.086
n1-highcpu-4	4	3.60GB	$0.160	$0.172
n1-highcpu-8	8	7.20GB	$0.320	$0.344
n1-highcpu-16	16	14.40GB	$0.640	$0.688

Shared Core

Machines for tasks that don't require a lot of resources but do have to remain online for long periods of time.

Instance type	Virtual Cores	Memory	Price (US$)/Hour (US hosted)	Price (US$)/Hour (Europe hosted)
f1-micro	1	0.60GB	$0.012	$0.013
g1-small	1	1.70GB	$0.032	$0.0347

US EUROPE / ASIA

-39.7% across the board, YoY

2016

Standard machine types

Machine type	Virtual CPUs	Memory	GCEU[1]	Lowest price[2] (USD) per hour with full sustained usage	Typical price[3] (USD) per hour	Full price[4] (USD) per hour without sustained use	Preemptible price[5] (USD) per hour
n1-standard-1	1	3.75GB	2.75	$0.035	$0.038	$0.050	$0.010
n1-standard-2	2	7.5GB	5.50	$0.070	$0.076	$0.100	$0.020
n1-standard-4	4	15GB	11	$0.140	$0.152	$0.200	$0.040
n1-standard-8	8	30GB	22	$0.280	$0.304	$0.400	$0.080
n1-standard-16	16	60GB	44	$0.560	$0.608	$0.800	$0.160
n1-standard-32[6]	32	120GB	88	$1.120	$1.216	$1.600	$0.320

US EUROPE / ASIA

-39.1% across the board, YoY

2016

Standard machine types

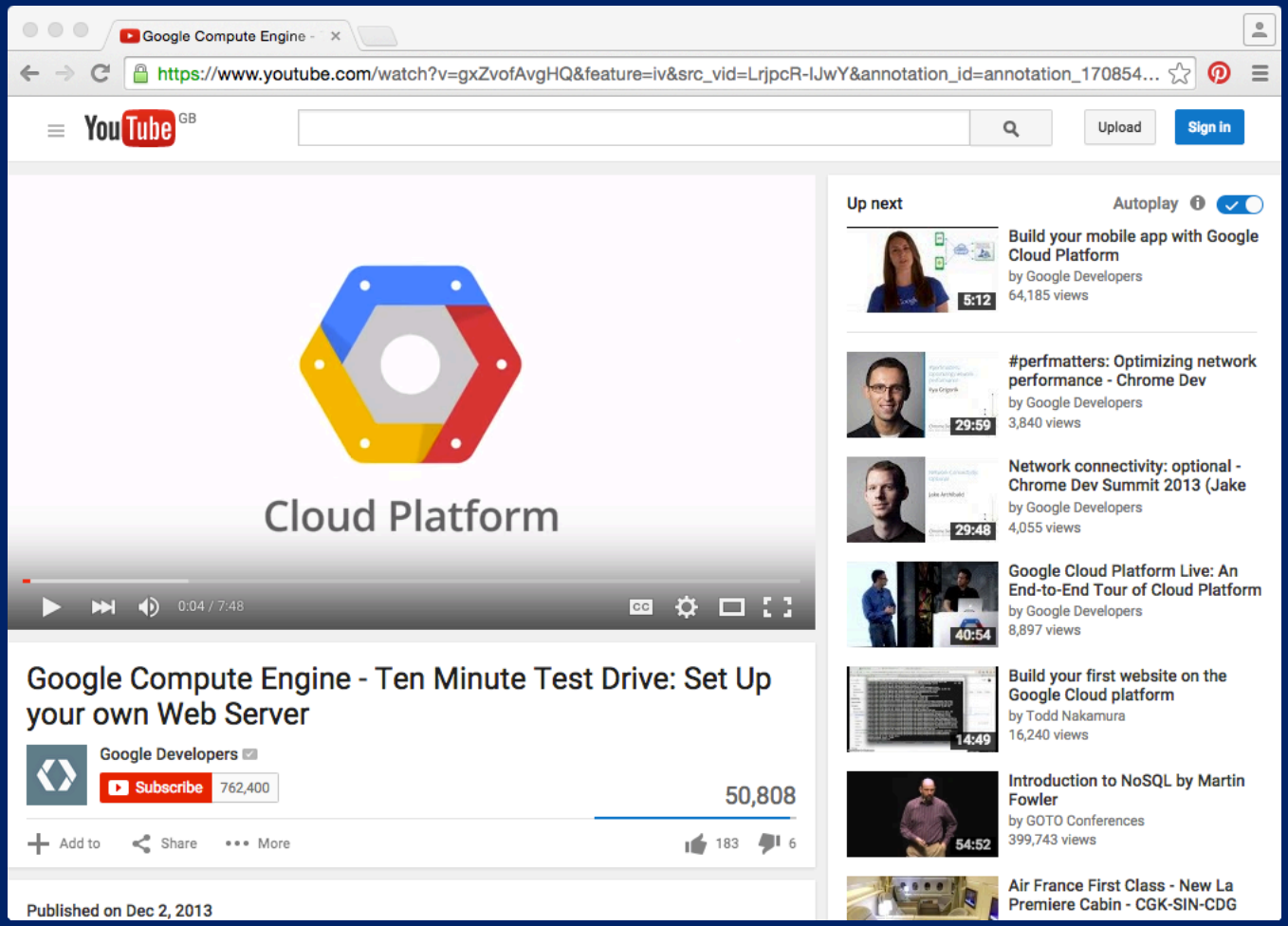
Machine type	Virtual CPUs	Memory	GCEU[1]	Lowest price[2] (USD) per hour with full sustained usage	Typical price[3] (USD) per hour	Full price[4] (USD) per hour without sustained use	Preemptible price[5] (USD) per hour
n1-standard-1	1	3.75GB	2.75	$0.039	$0.042	$0.055	$0.011
n1-standard-2	2	7.5GB	5.50	$0.078	$0.084	$0.110	$0.022
n1-standard-4	4	15GB	11	$0.156	$0.168	$0.220	$0.044
n1-standard-8	8	30GB	22	$0.312	$0.336	$0.440	$0.088
n1-standard-16	16	60GB	44	$0.624	$0.672	$0.880	$0.176
n1-standard-32[6]	32	120GB	88	$1.248	$1.344	$1.760	$0.352

2019

13% over 3 yrs: CAGR +4.16% across the board, YoY

Machine type	Virtual CPUs	Memory	Price (USD)		Preemptible price (USD)	
n1-standard-1	1	3.75GB	$0.0475	+13%	$0.0100	+$0.0055
n1-standard-2	2	7.5GB	$0.0950	+13%	$0.0200	+$0.011
n1-standard-4	4	15GB	$0.1900	+13%	$0.0400	+$0.022
n1-standard-8	8	30GB	$0.3800	+13%	$0.0800	+$0.044
n1-standard-16	16	60GB	$0.7600	+13%	$0.1600	+$0.088
n1-standard-32	32	120GB	$1.5200	+13%	$0.3200	+$0.176

One video… https://www.youtube.com/watch?v=gxZvofAvgHQ



Next…

Lecture 6: Virtualization. With Dan. Really with Dan.