A Cloud-based Application Storage Service for Ibis

Bas Boterman

June 25, 2009

Outline

Introduction

Project Details

Benchmarks

Applications

Conclusion

Background

- April 7, 2008, the Google App Engine, a platform for developing and hosting web applications, was introduced

Background

- April 7, 2008, the Google App Engine, a platform for developing and hosting web applications, was introduced
- To what extent can we use the Google App Engine for scientific purposes?

Background

- April 7, 2008, the Google App Engine, a platform for developing and hosting web applications, was introduced
- To what extent can we use the Google App Engine for scientific purposes?
- Robust distributed datastore with transactions and queries

Background

- April 7, 2008, the Google App Engine, a platform for developing and hosting web applications, was introduced
- To what extent can we use the Google App Engine for scientific purposes?
- Robust distributed datastore with transactions and queries
- This project is about designing an application storage server for Ibis

Thesis Title

A Cloud-based Application Storage Service for Ibis

Thesis Title

A **Cloud-based** Application Storage Service for Ibis

Thesis Title

A **Cloud-based** Application Storage Service for Ibis

> *"A style of computing in which dynamically scalable resources are provided as a service over the Internet."*

E.g. the Google App Engine.

Thesis Title

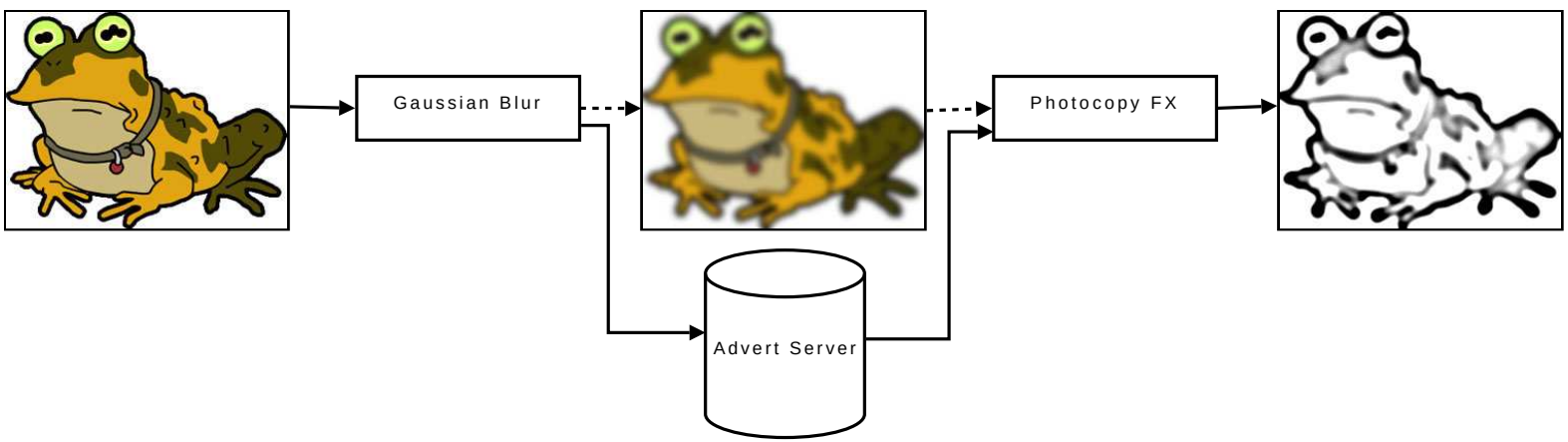
A Cloud-based **Application Storage Service** for Ibis

Thesis Title

A Cloud-based **Application Storage Service** for Ibis

> *"A service used by applications to store and retrieve application data."*

Example:



Thesis Title

A Cloud-based Application Storage Service for **Ibis**

Thesis Title

A Cloud-based Application Storage Service for **Ibis**

> *"The main goal of the Ibis project is to create an efficient Java-based platform for grid computing."*

- JavaGAT (Grid Application Toolkit)
- IPL (Ibis Portability Layer)

Google App Engine

"A platform for building and hosting web applications on the Google infrastructure."

- Python programming language
- Web application framework
- Authentication through Google Accounts
- 10 GB traffic per day
- 1 GB of data storage
- Free of charge.

(Dis)Advantages of the Google App Engine

Advantages:

- Powerful (distributed) database with query engine and transactions
- Replicating servers to improve scalability
- Authentication through Google Accounts
- Built-in frameworks (Django, WebOb, PyYAML)

(Dis)Advantages of the Google App Engine

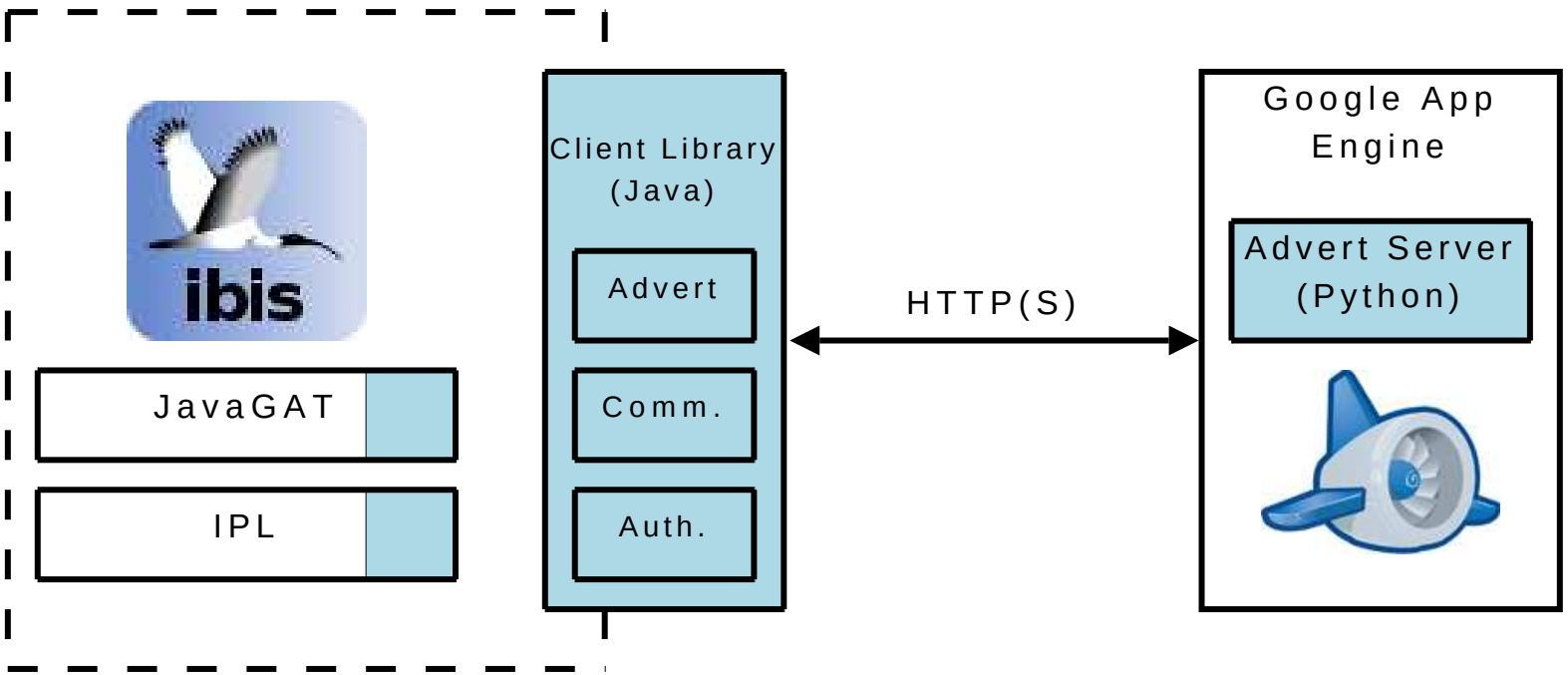
Advantages:

- Powerful (distributed) database with query engine and transactions
- Replicating servers to improve scalability
- Authentication through Google Accounts
- Built-in frameworks (Django, WebOb, PyYAML)

Disadvantages:

- Python 2.5 (limited functionality, compared to 3.0)
- HTTP(S) only
- Sandbox (no `fork()`, no threads, no file system)

Project Overview



Server Design

Public functions:

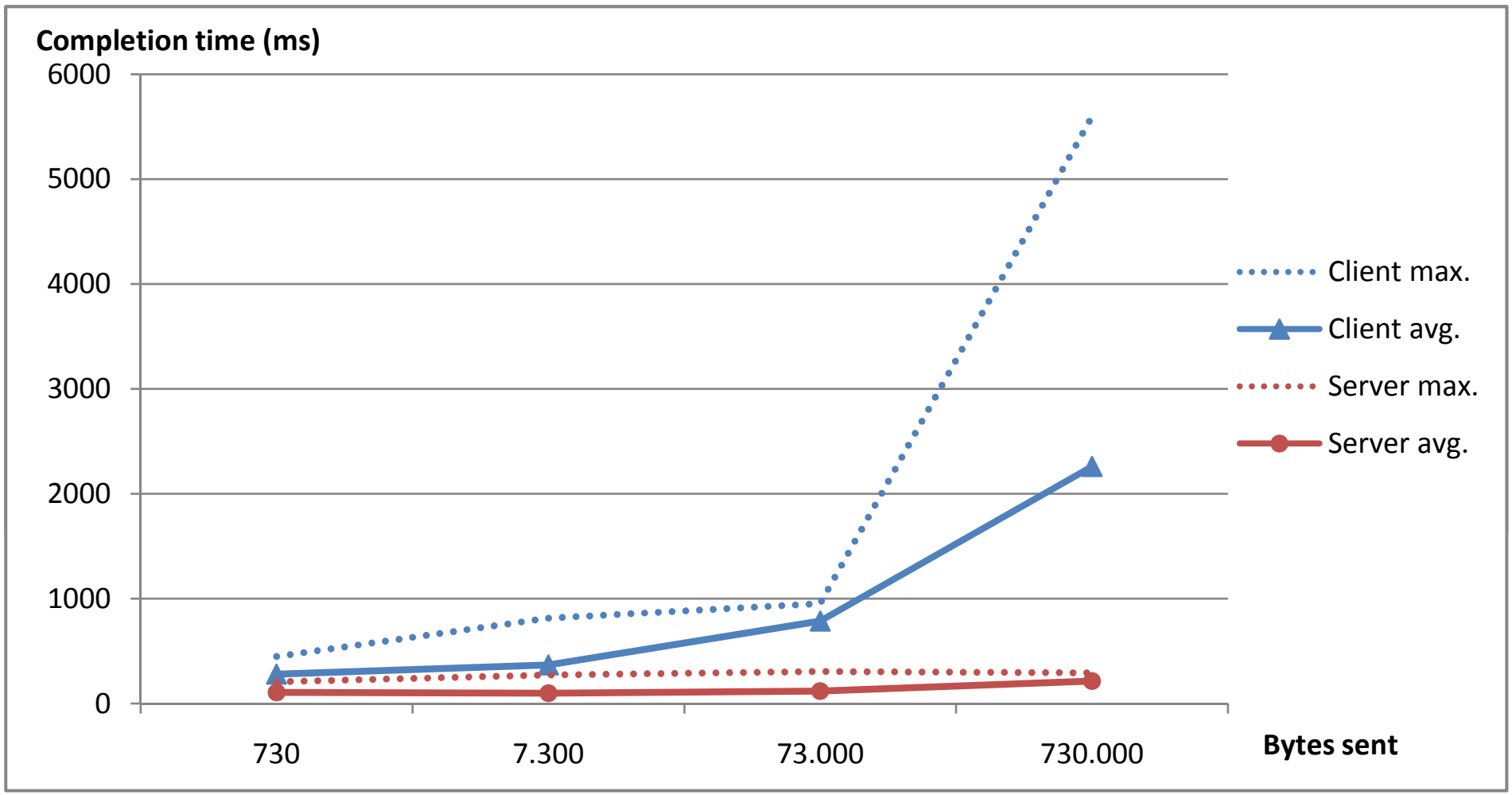
- `add(data, metadata, path)`
- `del(path)`
- `get(path)`
- `getmd(path)`
- `find(metadata)`

- `data`: Text
- `metadata`: Key-value pairs of Strings
- `path`: String

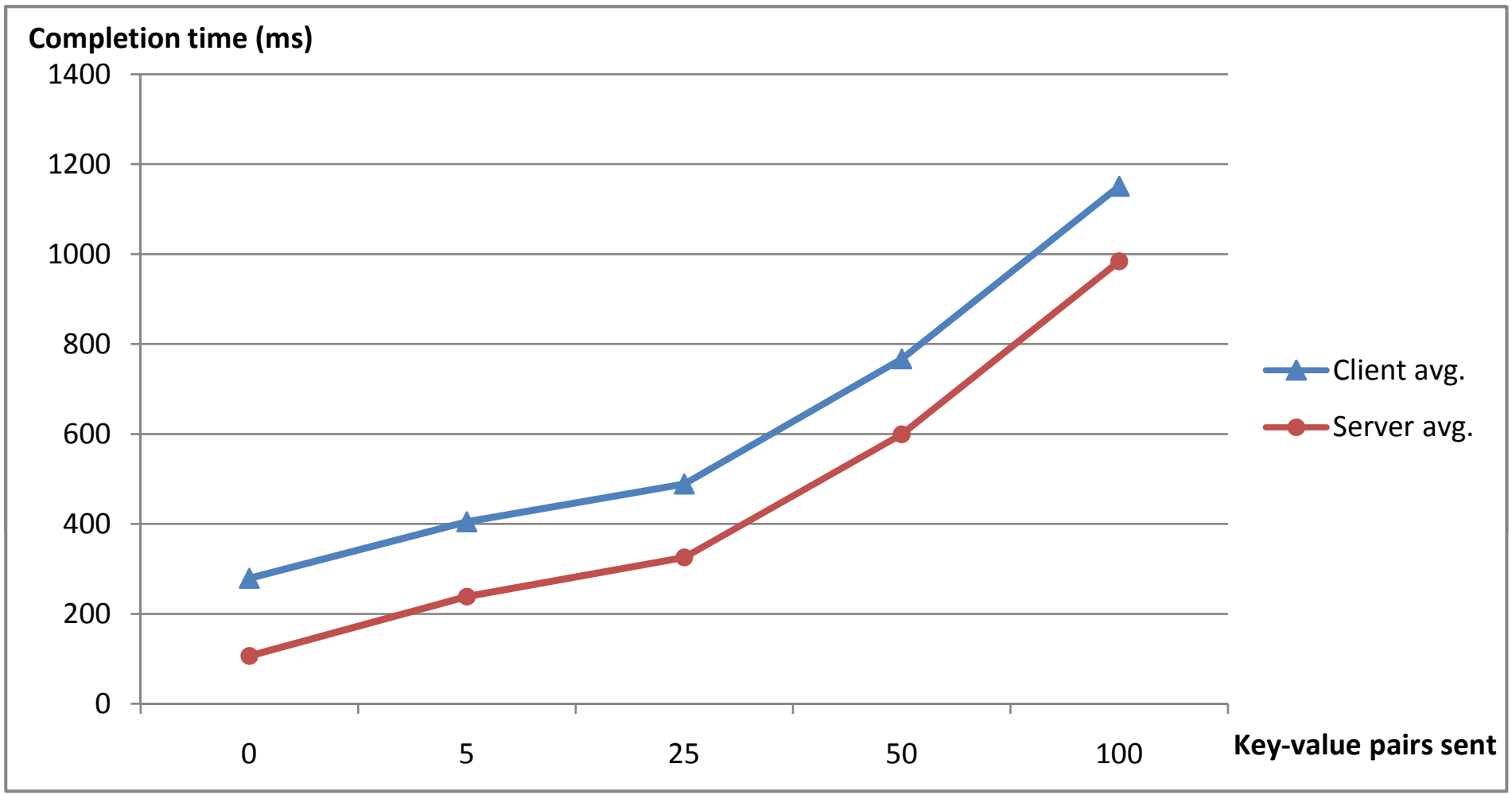
Client Library

- Written in Java
- Implements the Google App Engine Advert server's protocol
  - Base64 encoding
  - JSON encoding (XML alternative)
  - Error handling
- Establishes connections over HTTP(S)
- Takes care of authentication
  - Authentication through *Google ClientLogin*
  - Automatically refresh authentication cookie

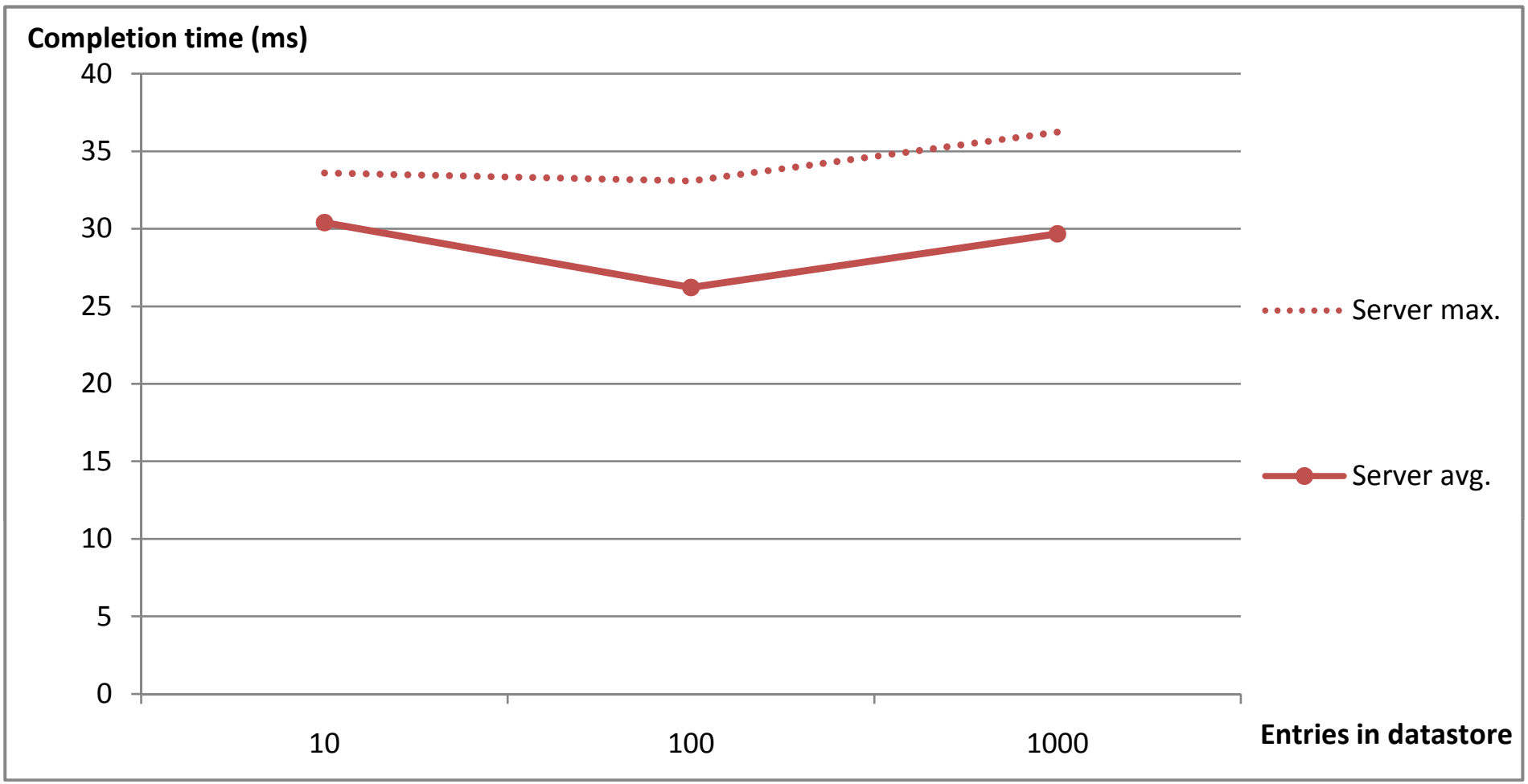
Adding Objects of Variable Size



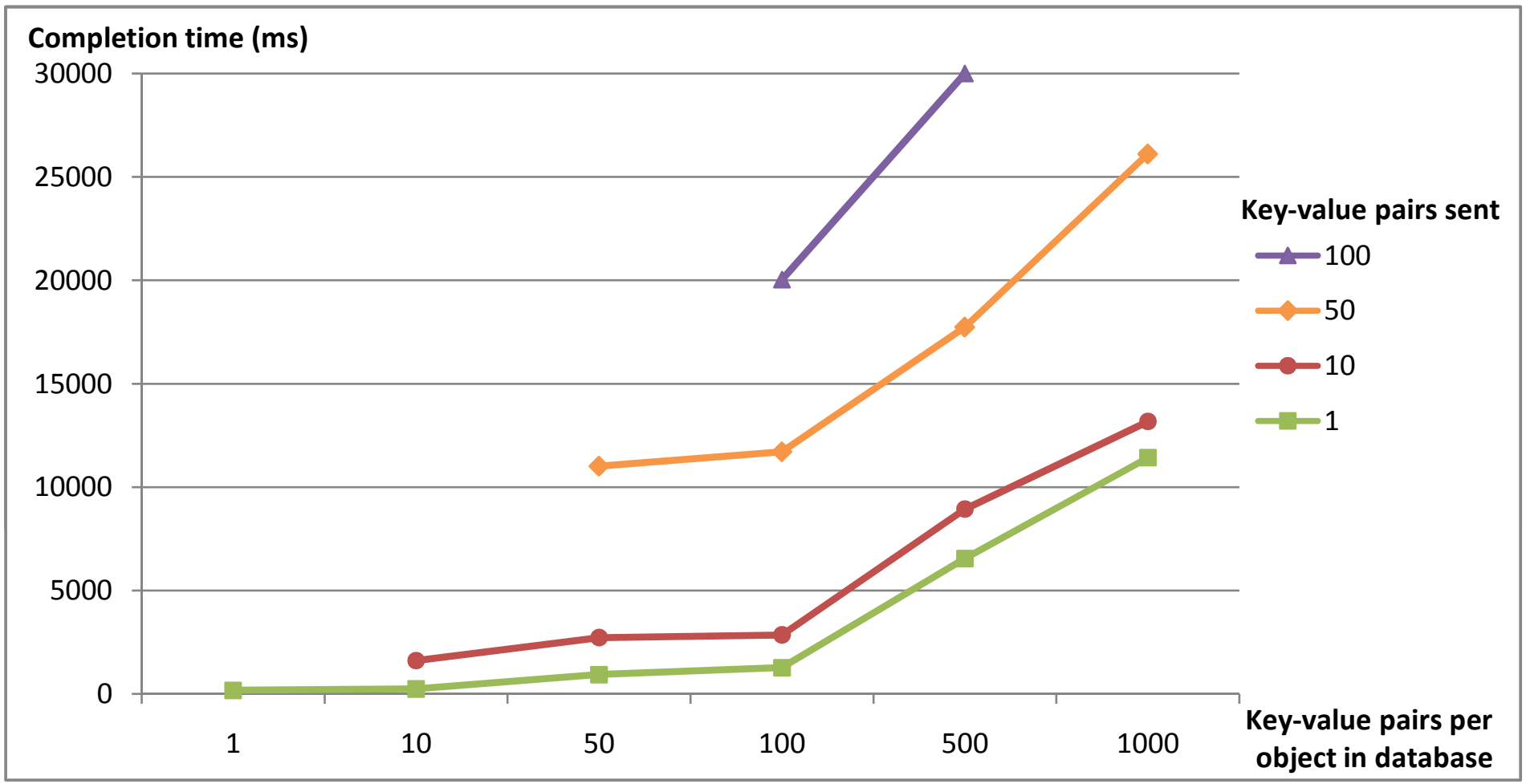
Adding Objects with Variable Meta Data



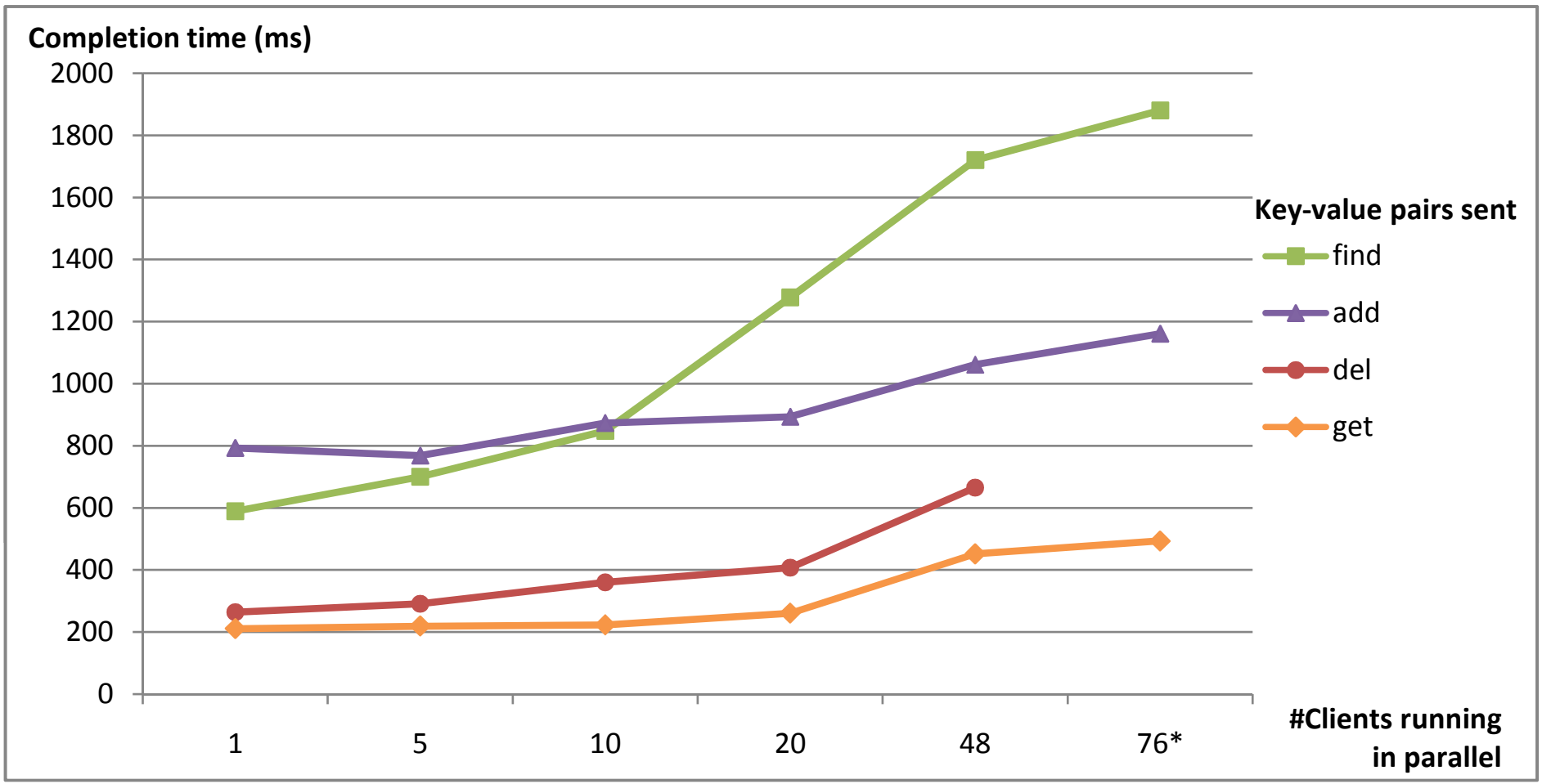
Retrieving Objects with Variable Meta Data



Finding Meta Data



Performing Operations in Parallel



Benchmark Evaluation

- Completion time depends on object size/meta data size, and on number of clients connecting simultaneously
- Completion time is independent on the datastore size
- Google has some restrictions
  - Response time cannot be over 30 seconds
  - Quota temporarily exceeds with requests larger than 5 MB
  - No more than 500 data items can be manipulated in one call
  - No more than 100 clients can use ClientLogin simultaneously

Outline

Introduction

Project Details

Benchmarks

Applications

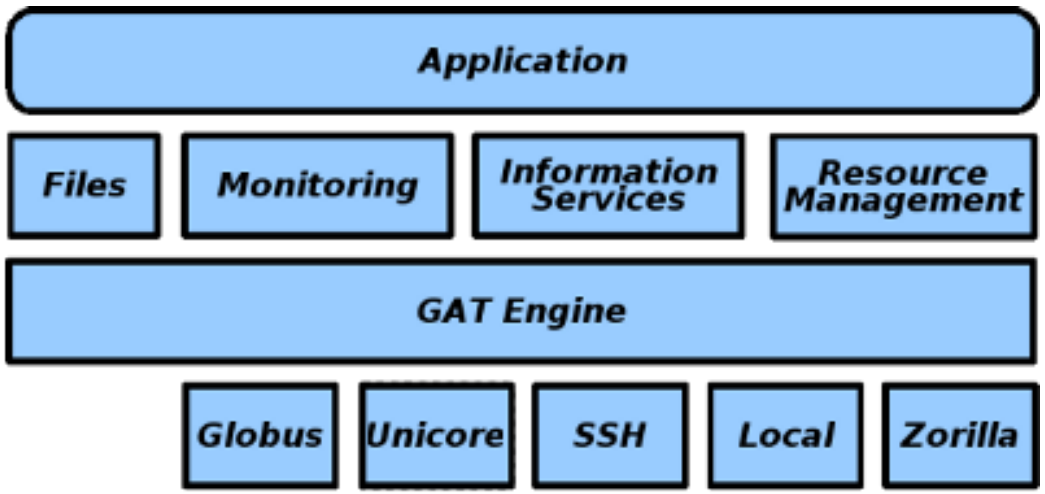
Conclusion

JavaGAT

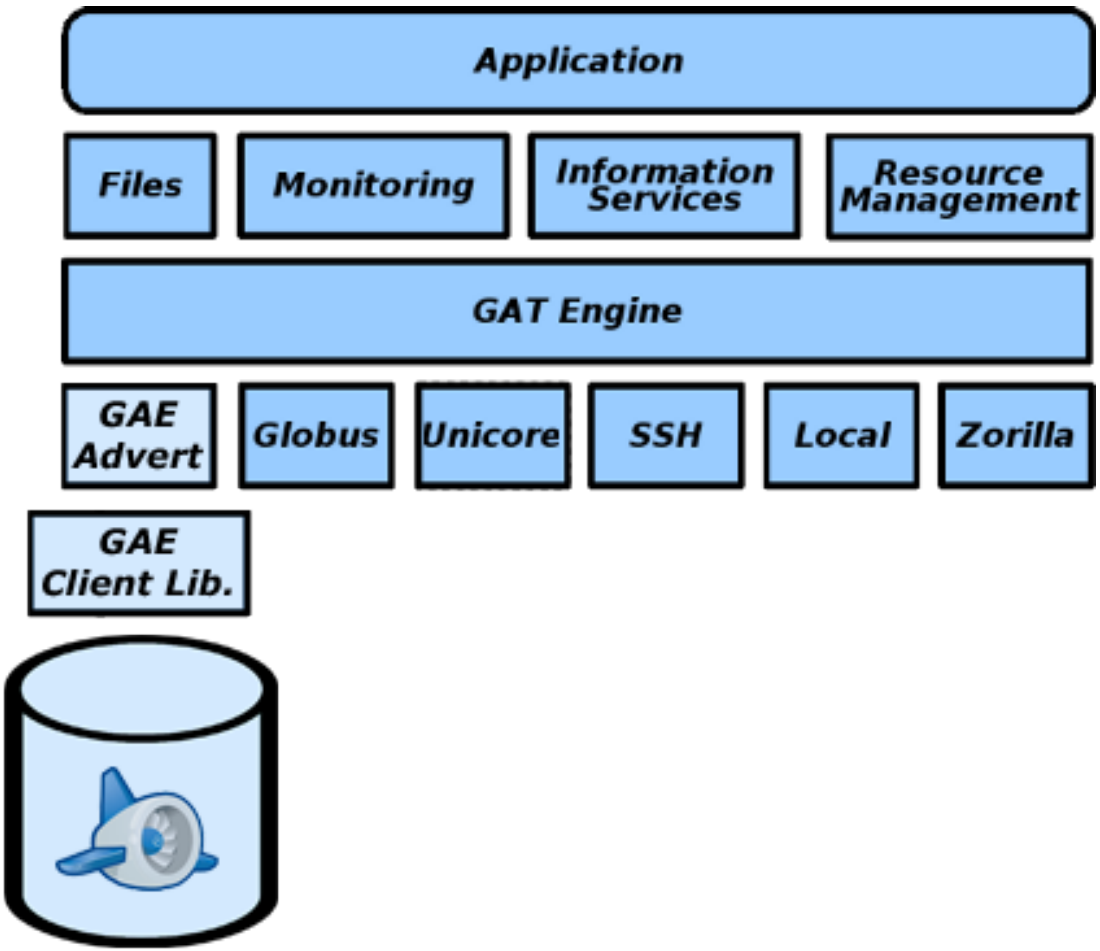
JavaGAT offers a set of coordinated, generic and flexible APIs for accessing grid services from application codes, portals, data managements systems, etc.

- File operations
- Job submission
- Monitoring
- Access to information services (AdvertService)

JavaGAT structure



JavaGAT structure with App Engine AdvertService Adaptor



App Engine AdvertService Adaptor

- So far, only a local AdvertService existed
- Scalable design (local AdvertService does not scale)
- Completely transparent to user
- Path handling is done locally

IPL

"The IPL is a communication library which is specifically designed for usage in a grid environment."

- Communication library for distributed applications
- All applications discover each other through an IPL registry server
- IPL registry server address needs to be passed to each Ibis application

IPL Example

```
$ $IPL_HOME/bin/ipl-server --events

Ibis server running on 130.37.20.18-8888~bbn230
List of Services:
    Bootstrap service on virtual port 303
    Central Registry service on virtual port 302
    Management service
Known hubs now: 130.37.20.18-8888~bbn230


$ $IPL_HOME/bin/ipl-run \
-Dibis.server.address=130.37.20.18-8888~bbn230 \
-Dibis.pool.name=test \
ibis.ipl.examples.Hello
```

IPL Example Using Advert Server

```
$ $IPL_HOME/bin/ipl-server --events \
--advert google://jondoe.appspot.com/identifier \
--user jondoe@gmail.com --pass north23AZ \
--metadata author=jondoe,created=24jun,pool=test,color=purple

Ibis server running on 130.37.20.18-8888~bbn230
List of Services:
    Bootstrap service on virtual port 303
    Central Registry service on virtual port 302
    Management service
Known hubs now: 130.37.20.18-8888~bbn230
```

IPL Example Using Advert Server

```
$ $IPL_HOME/bin/ipl-run \
-Dibis.advert.address=google://jondoe.appspot.com \
-Dibis.advert.username=jondoe \
-Dibis.advert.password=north23AZ \
-Dibis.advert.metadata=created=24jun,color=purple \
-Dibis.pool.name=test \
ibis.ipl.examples.Hello
```

IPL Server Bootstrap Mechanism

- No mechanism for bootstrapping IPL servers existed
- Large range of IPL servers can be started, whilst only remembering one (advert) address
- Different groups of Ibises can be started without (manually) having to configure each Ibis

Conclusion

Contributions:

- A Google App Engine Advert server, written in Python
- An Advert client library, written in Java
- An App Engine AdvertService adaptor for JavaGAT
- An IPL Server bootstrap mechanism

Conclusion

Contributions:

- A Google App Engine Advert server, written in Python
- An Advert client library, written in Java
- An App Engine AdvertService adaptor for JavaGAT
- An IPL Server bootstrap mechanism

Properties:

- Scalable application and datastore
- High bandwidth (until quota reached)
- No guarantees

Conclusion

Contributions:

- A Google App Engine Advert server, written in Python
- An Advert client library, written in Java
- An App Engine AdvertService adaptor for JavaGAT
- An IPL Server bootstrap mechanism

Properties:

- Scalable application and datastore
- High bandwidth (until quota reached)
- No guarantees

Future work:

- An App Engine Advert server, written in Java