SAGA Async Advert Adaptor

Hans Christian Wilhelm

December 15, 2011

Table of contents

Sync Advert Adaptor

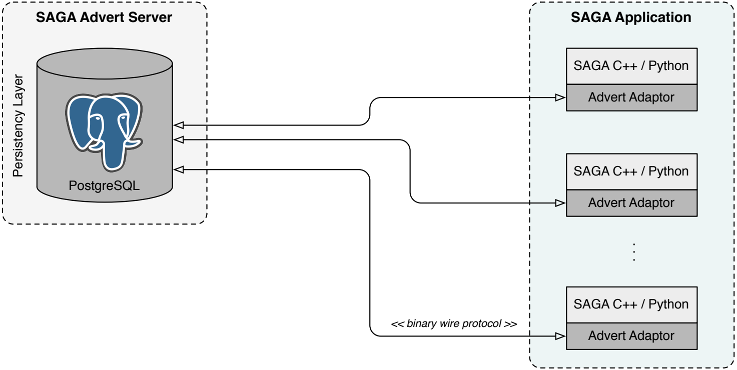

Figure: Default Advert and Fast Advert Adpator communication layout.

Sync Advert Adaptor

Communication layout

- Each Adaptor instance has a single connection to the database.
- Directories / Entries are opened with recursive queries.
- This leads to multiple Request/Response roundtrips.
- Each Attribute needs one query.
- This leads to one Request/Response roundtrip per attribute.
- Polling is needed to see if something has changed.

Fast Advert Adaptor

- Reduced query count.
- Still needs polling !

Async Advert Adaptor

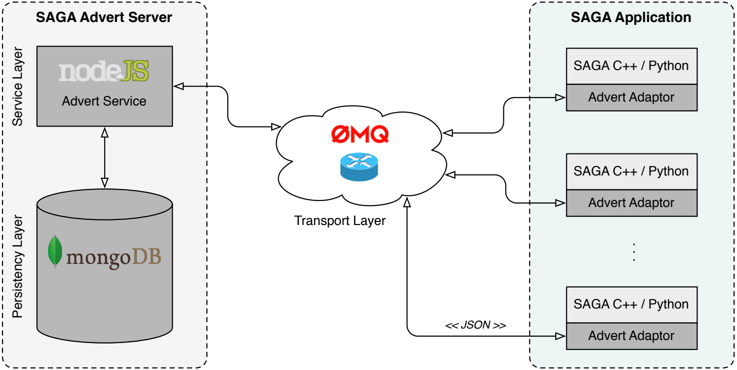

Figure: Async Advert Adpator communication layout.

Async Advert Adaptor

Communication layout

- Two ZMQ connections per Adaptor instance.
- One Request/Response connection to send commands.
- One Publish/Subscribe connection to receive notifications.
- No recursive database layout.
- Directories / Entries are transported as a union.

Roundtrips

- Only one roundtrip to open an Diretory / Entry.
- No extra roundtrip to query an attribute.
- No polling to the server.

Architecture

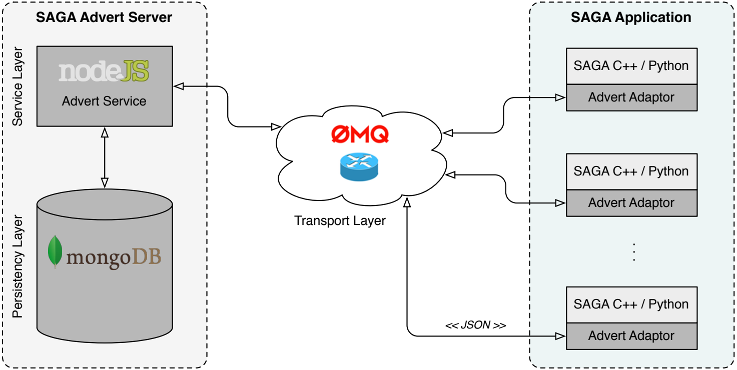

Figure: Async Advert Adpator Architecture

Technology

NodeJS

- The Advert Server is implemented in the non-blocking JavaScript framework NodeJS.
- Manages incoming connections.
- Publishes messages to the Advert Adaptor client.
- Communicates with the Database.

MongoDB

- MongoDB is used as the Database persitency layer.
- Document based, each directory/entry modeled as document.

Technology

ZeroMQ

- ZeroMQ sockets connect the Advert Adaptor to the server.
- REQUEST/RESPONSE socket used so send commands to the server.
- PUBLISH/SUBSCRIBE socket used to notify the Advert Adaptor clients.

JSON

- JSON is used as a lightweight data interchange format.
- The Advert Addaptor sends JSON coded commands e.g. Open, Close, Create to the Server.
- The server responds with a JSON coded diretory / entry.
- The server publishes JSON coded document ID's to the Advert Adaptor Clients.

Technology

Links

- http://nodejs.org/
- http://www.mongodb.org/
- http://www.zeromq.org/
- http://www.json.org/
- https://github.com/JustinTulloss/zeromq.node
- https://github.com/LearnBoost/mongoose

Howto install the Async Advert Adaptor

SAGA Core

First make sure you have a fully working SAGA Core installation on your system ! Installation details can be found on http://www.saga-project.org/documentation/installation

Howto install the Async Advert Adaptor

Get ZeroMQ

Befor the installation of the Async Advert Adaptor we need to install ZeroMQ (Release 2.1). Download the POSIX tarball from http://www.zeromq.org/intro:get-the-software and install it.

Install ZeroMQ

```
tar xvzf zeromq-2.1.10.tar.gz
cd zeromq-2.1.10
./configure –prefix=/choose/your/install/path/zeromq-2.1.10
make
make install
```

Howto install the Async Advert Adaptor

Get AsyncAdvertAdaptor

After the installation of ZeroMQ it is time to get the Async Advert Adpator from the SVN repository.

Install Async Advert Adaptor

```
svn co https://svn.cct.lsu.edu/repos/saga-adaptors/async_advert_adaptor
cd async_advert_adaptor
./configure –with_zmq=/install/path/zeromq-2.1.10
make
make install
```

Testing environment

Testing Server

- Async Advert testing server gw68.quarry.iu.teragrid.org
- sqlasyncadvert://gw68.quarry.iu.teragrid.org/
- Ports are hardcoded at the moment ! (5557/5558)

Getting started

- Go to your SAGA Core install path /bin
- Play around with the saga-advert-... commands.
- Try the Async Advert Adaptor in your own projects.

Example

```
./saga-advert-dump-directory
sqlasyncadvert://gw68.quarry.iu.teragrid.org/
```

Optional Benchmark Tool

Python Benchmark Tool

- Install the SAGA Python bindings.
- Checkout the benckmark tool from https://svn.cct.lsu.edu/repos/saga-adaptors/async_advert_adaptor/benchmark/
- Have a look at the README file and start testing.

Example

- svn co https://svn.cct.lsu.edu/repos/saga-adaptors/async_advert_adaptor/benchmark/
- cd benchmark
- python advert-benchmark.py sqlasyncadvert://gw68.quarry.iu.teragrid.org/your/benchmark/dir -p -c 10 -a 10 -i5

Effective usage

Directories / Entries are transported as unions

Keep in mind that directories / entries are transported as unions over the wire. That means all attributes and vector attributes are transmitted with every directory / entry.

Usage hints

- Try to keep attribute count per directoy / entry as low as possible.
- Don't use a single entry to coordinate all workers.
- Assign every worker to it's own entry.
- Try to keep entry count per directory as low as possible.
- Use different directories to model dependancies.