AMP 3.0 Technical Overview

Contents

System Requirements

RAM	8 GB minimum, 16GB recommended
CPU	QuadCore 3.4GHz 64bit
HDD	300 GB
OS	Debian, Scientific Linux, Windows Server 2008+ *Also other well-supported/secure operating systems like CentOS. Latest stable version with long support is recommended for new server installations.*
Networking	public IP, domain name; HTTP 80, SSH ports open
ArcGIS	minimum 50GB HDD with a dedicated partition (e.g. /opt/arcgis), opened 6080 port
Email Alerts	SMTP port open

AMP Production Setup

AMP 3.0 is deployed using Apache Tomcat 7, an open-source Web Server that implements several Java EE specifications. Since 2.12 release, AMP is based on Java 8. The open-source Apache HTTP Server is used to deliver the application content to the country specific domain, with the right pointers to the Public Portal and AMP itself. AMP benefits from its security, traffic control efficiency and its compliance with current HTTP standards. AMP setup additionally uses some tools like Maven 3.3.9 and gitHub. The AMP Public Portal is deployed as a separate application and can even run on a different machine if necessary. The Public Portal uses Drupal 7 which runs with PHP5 and has its own DB instance for its settings and storage.

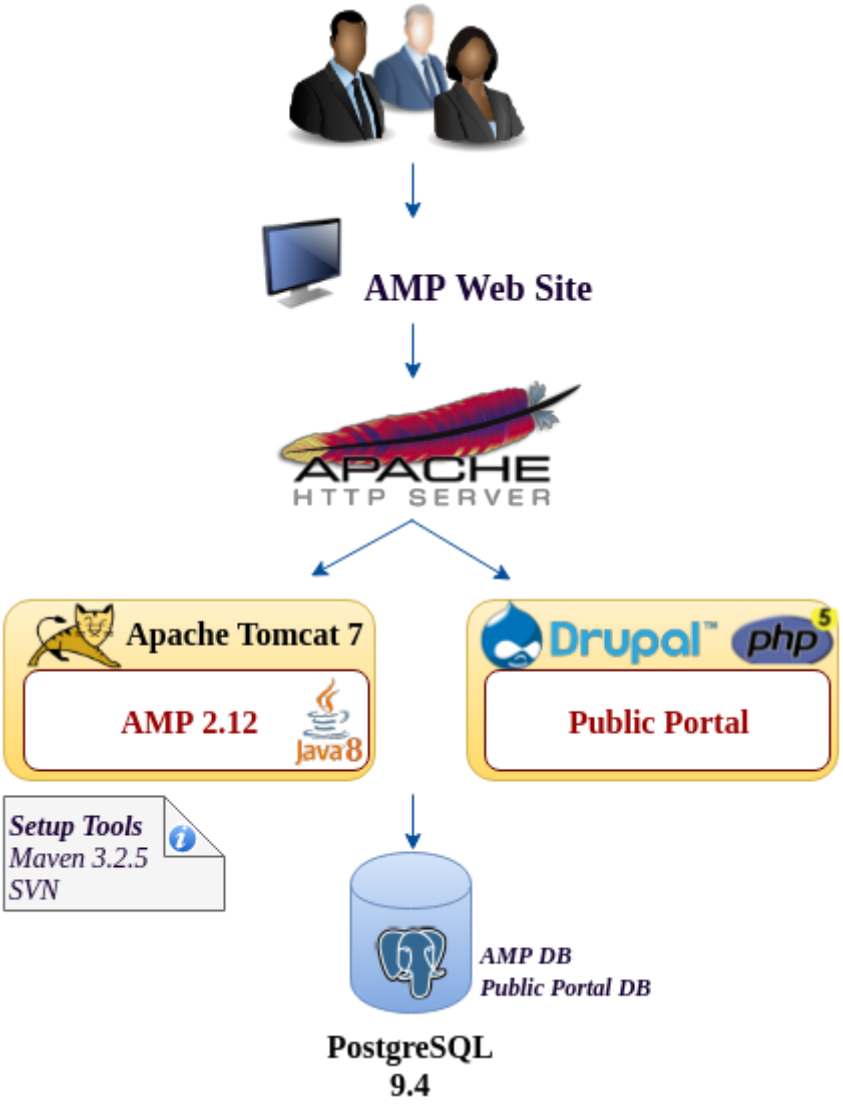

Figure 0 - AMP Setup

For each installation, the AMP Team provides a detailed AMP Installation Guide about how to setup and configure all required software. This guide also provides a set of scripts and procedures to handle the automatic DB backup and logs rotation. Additionally the team has developed an AMP Troubleshooting Guide that provides the procedures that the local IT department can follow.

Architecture

The history of AMP starts in 2005. During more than 10 years of continuous growth, the platform has delivered to its users a large amount of versatile, highly detailed and configurable features. Over this significant (in IT terms) amount of time, the technology has evolved as well. AMP always strives to keep up to date to the latest technologies and the team is choosing wisely the most important modules to improve both User Interface & Experience, as well as the underlying algorithms and architecture, that also targets a significant reduction in maintainability costs. This also allow handover of the development and maintenance to any other third party developers which have been hired by our partner governments.

AMP stack currently mixes newer, more innovative as well as more established technologies. In the diagram

below you can find a high level overview.

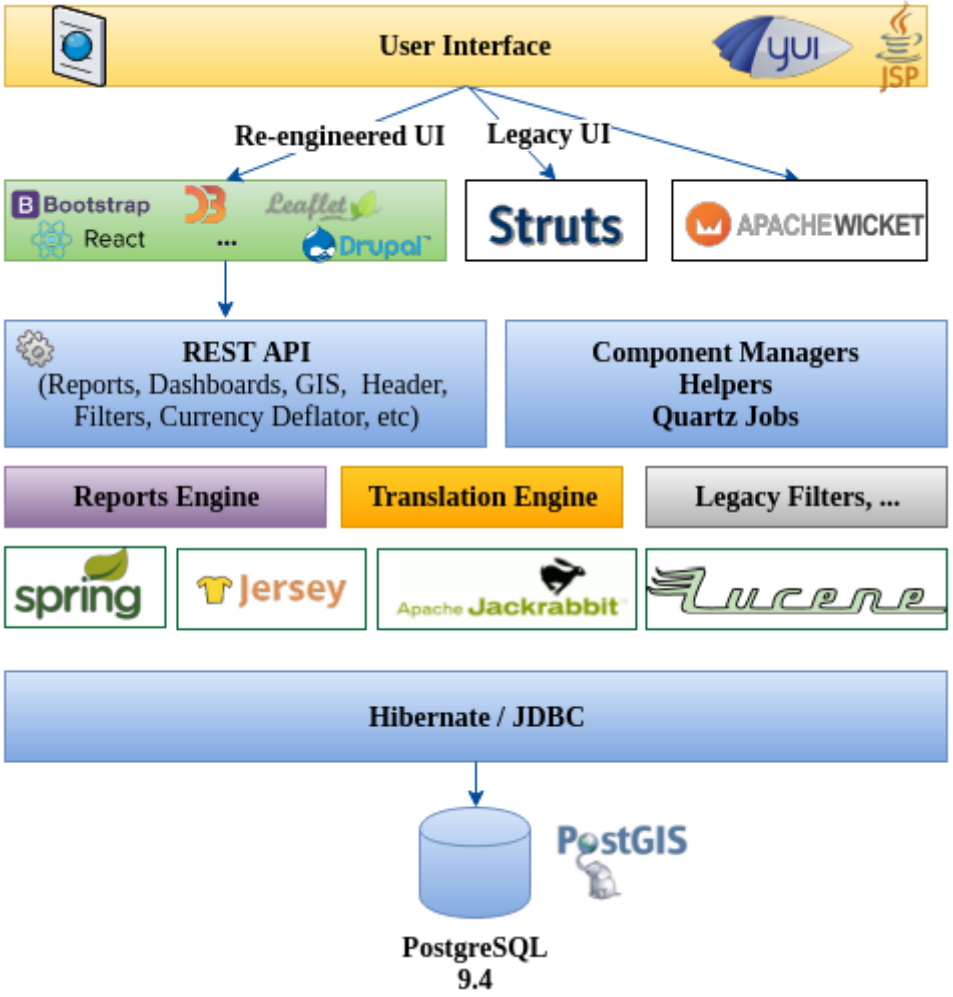

Figure 2 - AMP Overview

In the recent 2.10, 2.11 and 2.12 releases, AMP team focused to reengineer some of the most critical AMP modules, along continuous new client features development. Below is a zoomed in view of the 2.10-2.12 Reengineering achievements.

Figure 3 - Reengineering Updates

The first major step of the reengineering was to decouple some UI components from the backend by introducing AMP Restful API. This allowed an independent UI development with relatively independent technologies stacks based on needs.

One of the major achievements is streaming all Reporting based components (Reports, Tabs, Dashboards, GIS, Public Portal) through a common Reports Engine API. Following the reusability concept, AMP team built also a common Filters Widget on top of Filters API. Import / Export of IATI activities now are handled by an independent IATI Importer tool integrated with AMP via Activities API. Public Portal could evolve as a separate application integrated through a Public AMP API to query and display data.

The UI of all reporting based components was updated to the latest technologies, which allows an easier maintainability and more versatile set of functionalities.

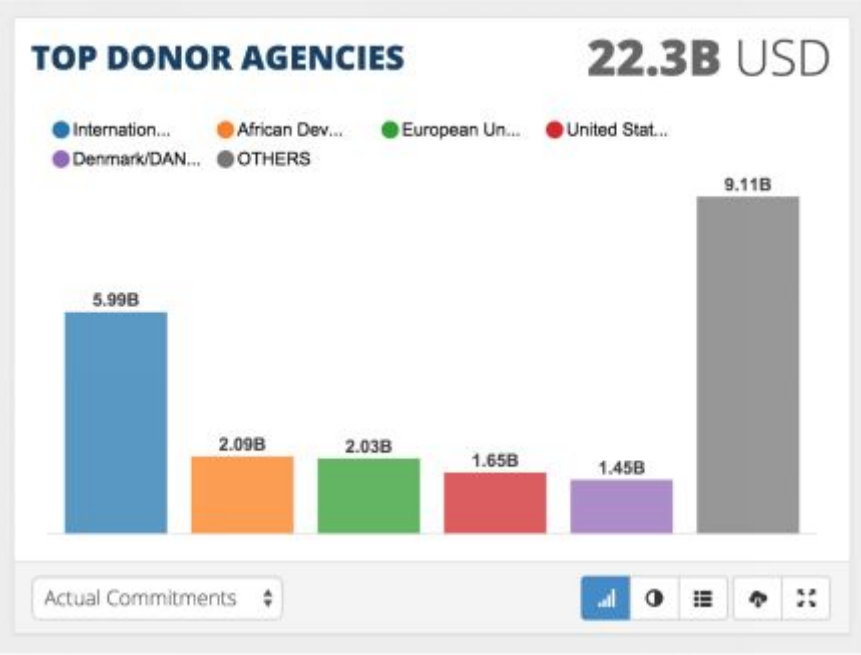

Figure 4 - Dashboards - Top Donor Agencies Chart

Technologies

DG leverages Open Source technologies both as part of the end-product applications, as well as its supporting tools in the Software Development process. Our Aid Management Platform relies upon Jenkins for continuous integration. DG uses both SVN and Github extensively in its development process, and our developers are active participants in the Open Source community, with experience in all common Open Source practices.

Datastore

AMP system relies on an open source PostgreSQL Database. It is well known for its high performance, reliability and unlike some other open source RDBMS, it provides a PostGIS extension that we use for the GIS module. PostgreSQL fulfilled AMP system needs and proved stable over an extensive period of usage on different environments with different resources. We have defined a set of configurations to tune PostgreSQL for AMP needs to gain the best performance for each country setup. As part of AMP 2.12 development, we upgraded to **PostgreSQL 9.4** to embrace its new improvements and features. NiReports Engine was the first component to benefit from its JSON support - used to store reports generation statistics.

Document Management is handled by the open source content repository for the Java platform Apache Jackrabbit. This feature rich platform developed since 2004 is in continuous development.

Backend

The Aid Management Platform 3.0 backend runs on the latest Java 8 release. Java 8 includes new features like Lambda Expressions, new Date/Time API, etc that were essential for the reengineering of the Reports Engine. The Java-based platform brings us the multitude of Java frameworks, cross platform capability, a large pool of technical knowledge shared on Internet and hard copy books, as well the possibility to handover to any other

third party Java experienced developers.

AMP uses Hibernate 4.x core and Hibernate 2.x for Entities mappings to RDBS DB. Using Java Persistence API allows us to switch to a different RDBS, though PostgreSQL proved to be the right long term choice for us. However we plan to switch to more recent Hibernate version for a better set of features that the developer team would be able to embrace.

The open source Spring 3.x Security framework is used to provide AMP authentication and authorization with its protection against attacks like session fixation, clickjacking, cross site request forgery, etc.

Apache Lucene open source search software enables Java-based indexing and advanced search capabilities among AMP data like activities, reports, documents, etc.

Struts 1.x Forms and Actions is used in our legacy sections of the application.

For the complex features of the Activity Form, we utilize Apache Wicket open source, component oriented, serverside, Java web application framework. Invented in 2004, it is one of the few serverside web frameworks that continues to be in use today and still proves to be one of the best choices for special needs.

AMP API was developed on top of Jersey 1.x RESTful Web Service open source framework for Java that provides support for JAX-RS APIs and servers as JAX-RS (JSR 311 & JSR 339) Reference Implementation.

For various automated Jobs like Auto closing activities, alerts notifications, etc, AMP relies upon Quartz Job scheduling open source library. It allows us to have a very grained configuration of the schedules and run multiple jobs concurrently.

To provide AMP with a rich set of features and capabilities, other libraries are used to deliver such functionality like Export to MS Word, MS Excel via Apache Poi, logging system events for tracing & debugging capabilities via different trace levels of Log4j, etc.

Frontend

AMP evolved in a context of mid 2000' technologies, revolving periodically to more modern ones. The system though is quite big, and amid AMP growth and periodic reengineering we have some legacy areas. Legacy front-end is based on JSP with yUI, jQuery and JS scripts.

AMP 2.10-2.12 Reengineering efforts revamped most relevant modules UI like GIS, Dashboards, Tabs, Filters, some Admin parts, etc. that embrace now a new look and feel. Modern JS libraries and tools are used to ease the development and maintenance. The decoupling from AMP system via its API, allowed us to address each AMP module's particular needs.

Figure 5 - Frontend Technologies

Charts are built with help of D3.js, a JavaScript library for data-driven documents. D3.js combines powerful visualization components with a data-driven approach to DOM manipulation, with an emphasis on web standards. We also reuse the pre-build D3 components provided by NVD3.

We reuse free and open-source collection of tools provided by Bootstrap, that includes HTML and CSS-based templates for forms, buttons, navigation and other interface components, as well as optional JavaScript extensions.

Backbone allowed us a quicker development of models with its support for key-value binding, custom events, collections with a rich API of enumerable functions, views with declarative event handling, that could be easily connected to AMP API. These contributed to an easy development of different elements for Reports UI.

Open source jqGrid plugin on top of jQuery came in to support the Tabs results display in a simple gridview, with its support for pagination, AJAX, etc.

Using the server-side rendering capabilities of React, the application can serve pre-rendered HTML with fully-functioning links for dynamic sections of the site. This combination yields a fast user experience, full accessibility, and visibility to search engines. Using React to separate application state from View code makes

development predictable, with a low rate of defects.

To provide interactive GIS experience, AMP relied upon Leaflet and Esri-Leaflet plugins. Built with simplicity, performance and usability in mind, it enabled us with a full set of mapping features, as well as mobile friendly support. Esri-Leaflet provided us with tools for consuming ArcGIS services with Leaflet.
Public portal is built on top of Drupal 7 from which a variety of libraries for different CMS needs are used.

Various AMP front-end components benefit from the feature-rich set of JQuery utility methods for HTML document traversal and manipulation, event handling, etc. Other useful functional programming helpers from Underscore allowed AMP team to get just another opportunity to focus more on the actual features development.

The API decoupling allowed UI components to be developed without AMP necessarily running on the front-end developers' machines. You may use some mock JSON as input/output or link to a remote dev server with API. A set of open-source JS development tools enables the team for this independent development. Both locally and in production, Node.js provides a JavaScript run-time environment. It uses an event-driven, non-blocking I/O model that makes it lightweight and efficient. Node.js' package ecosystem, npm, is the largest ecosystem of open source libraries in the world. Browserify framework allowed some modules dependencies to be required in the browser as needed.

Reengineering

AMP 2.10/2.11

AMP API

The AMP API brings a radical change in AMP evolution. Modern Web / Mobile Applications strive to have their frontend and backend decoupled from one another. AMP has stepped in to align to most modern Web Application principals to integrate with AMP components and external clients through a RESTful API. That enabled us to follow a modular development approach, providing freedom to fron-end team for libraries choice that can best suit each component needs.

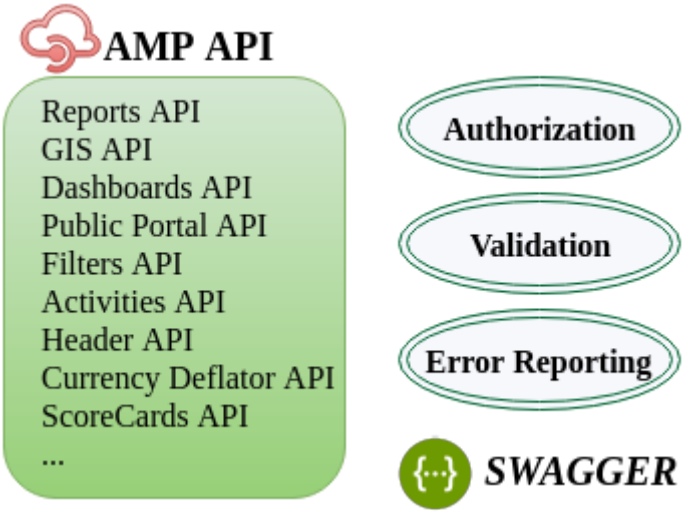

Figure 6 - AMP API

Developing of a new or Rre-engineeringenginerring of an existing AMP module from now on implies building up its special API. We also have a set of common API (like Reports, Filters, Settings API) that are

reusable among components. Swagger Tool has been integrated into AMP and can be used directly from a running AMP to check the latest API. It also saves on team effort to document the API in code (for developers) that can be also accessed by QA or external clients.
The AMP API has been enhanced to deliver its data only for authorized requests. Special user security checks and API token integration has been implemented for Activities API as part of integration with IATI Importer Tool. AMP API data input is validated and and errors are delivered to the REST client. We are still in progress of iterative development of these and other common features across different API components.

New Look and Feel

A significant effort has been put to provide a modern looking and user friendly design for some of the most frequently used components, as well as new one that were added during these releases. Previously outlined front-end technologies facilitated the team to focus on the actual features delivery and performance.

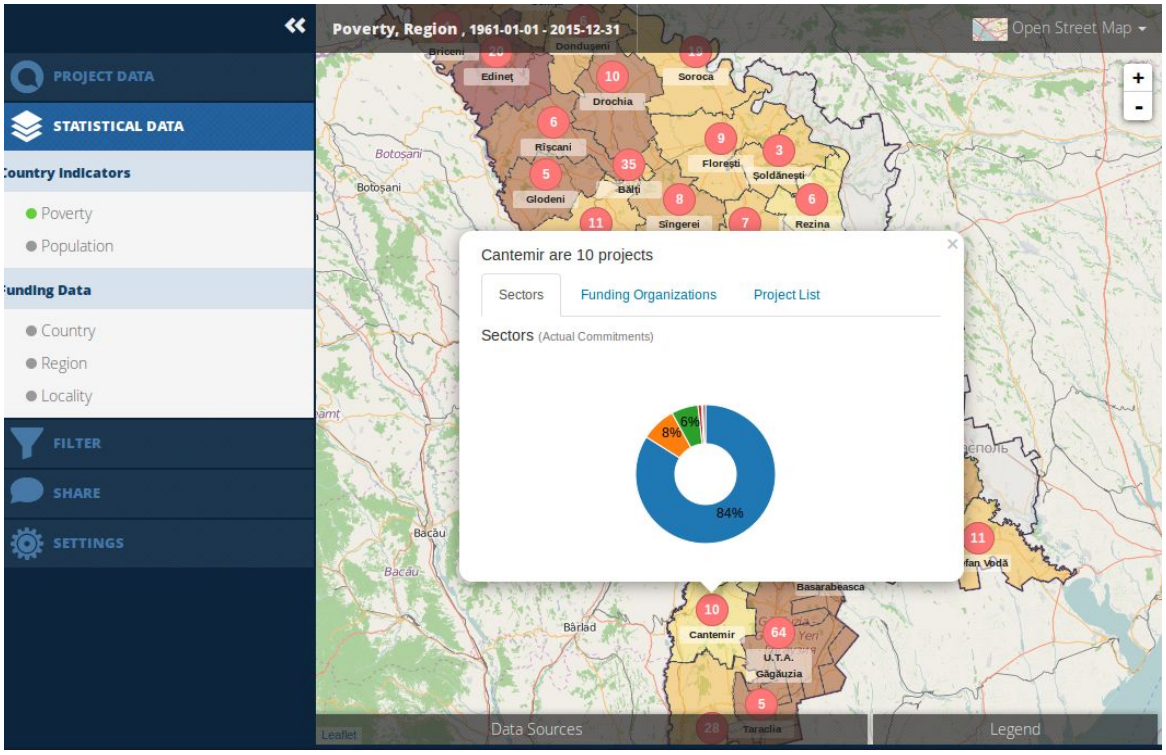

Figure 7 - AMP GIS

Users also benefit from consistent experience when using common components like Filters.

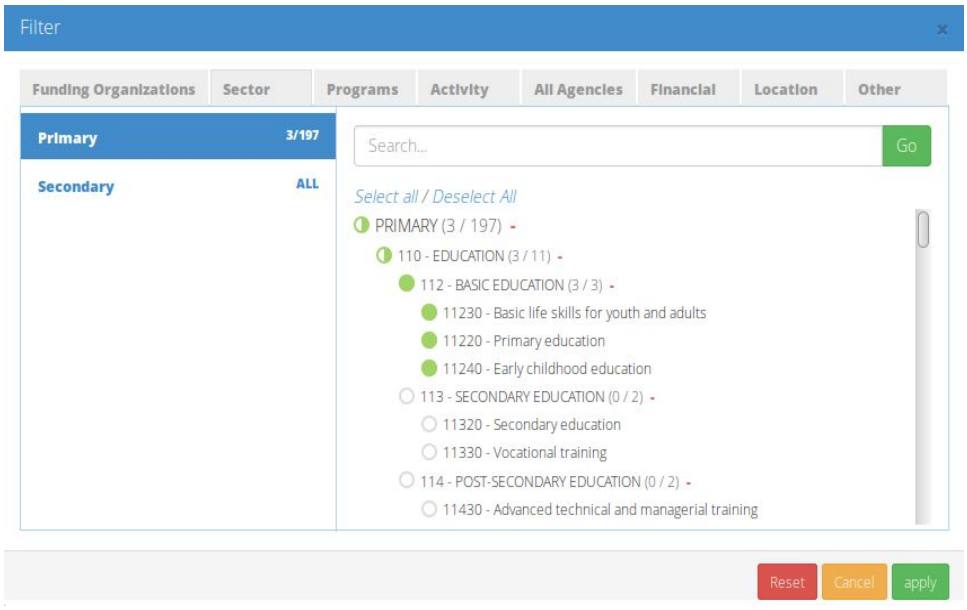

Figure 8 - AMP New Filters Widget

The API decoupling also enabled us to deliver AMP Public Portal as an external application running on top of Drupal 7 for its CMS features needed here.

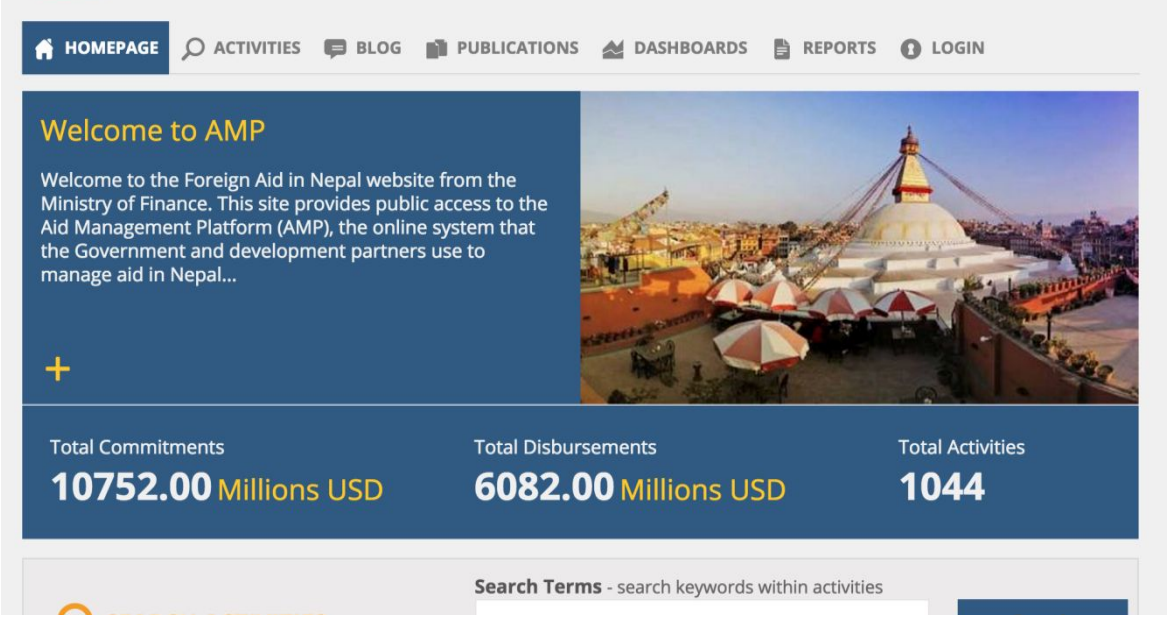

Figure 9 - AMP Public Portal

IATI Importer upgraded AMP possibilities to process the latest IATI 2.X standards, while bringing a more modern look to the user.

AMP 2.12

2.12 brings in another critical change – an updated report engine. This engine came in to fulfill some very specific needs of AMP users, that a generic report engine is not built for. Since 2.10+ we have utilized a Mondrian-based analytical reporting tool. Mondrian has proven to be not fully appropriate solution for our users' needs, since it focuses more on a business analysis by providing aggregated data that can be drilled down consecutively. AMP users on another hand frequently needs detailed reports from the first report load. The AMP 2.12 Reports engine was designed to consider all special aspects accumulated over the many years

of AMP Reporting.

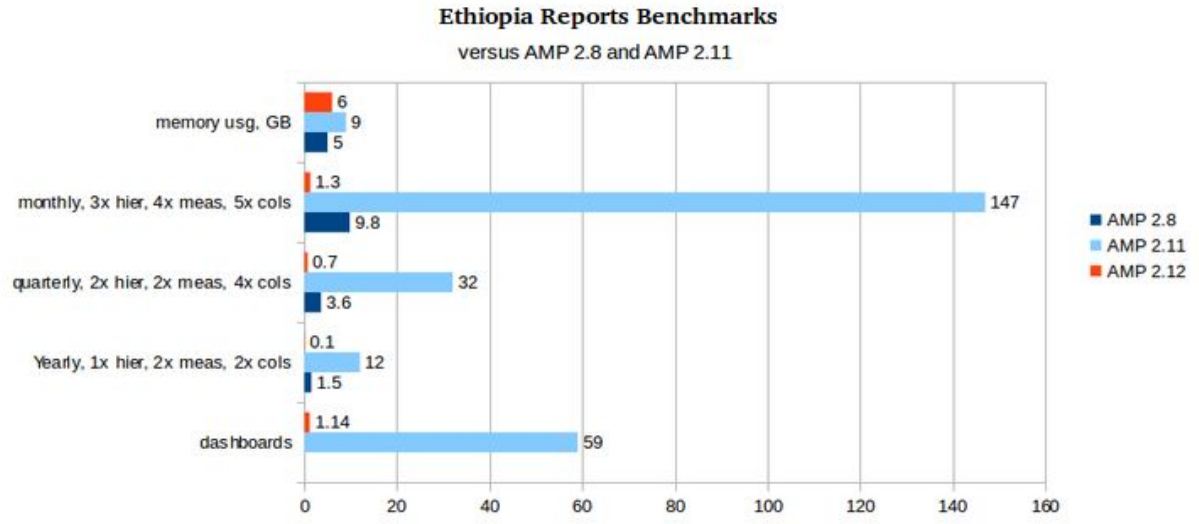

Figure 10 - Ethiopia Reports Benchmarks

The AMP 2.12 reports engine was built with several goals in mind. It has a core component that can be reused by other reporting applications. It can deliver from very simple to pretty complex reports in quite small amount of time. The core component stays untouched, configured with application specific schema. Additional custom measures, with specific logic, are easy to implement on top of that.

Figure 11 – AMP 2.12 Reports Architecture Overview

AMP's recently developed API layer allows re-engineered reporting modules to stay intact while the report engine was fully replaced. Very few API improvements have been added to complement some speed improvements. One of them was to reduce the network traffic as shown in the image below.

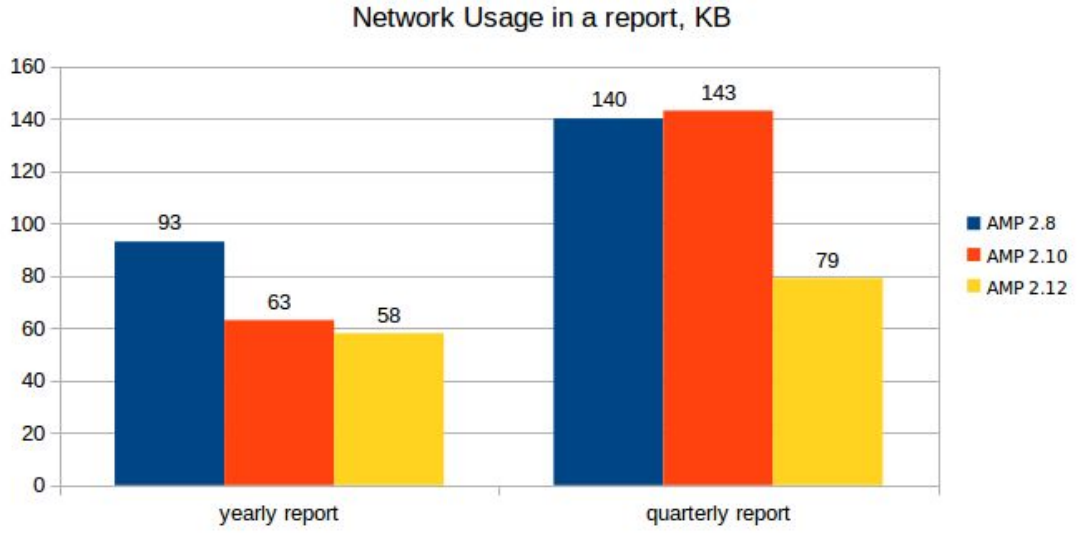

Figure 12 - Network Usage

Reports UI has been refactored to improve some data presentation speed, by delaying filters load and simplifying the Saiku UI that it is built on top. Some of its features are not applicable to AMP, hence some unused libraries now are not loaded.

AMP 2.12 brings AMP users high speed reporting. This improvement affects not only AMP Reports, but also all other modules that rely upon them. For instance dashboards load much more quickly in 2.12 than in previous versions.

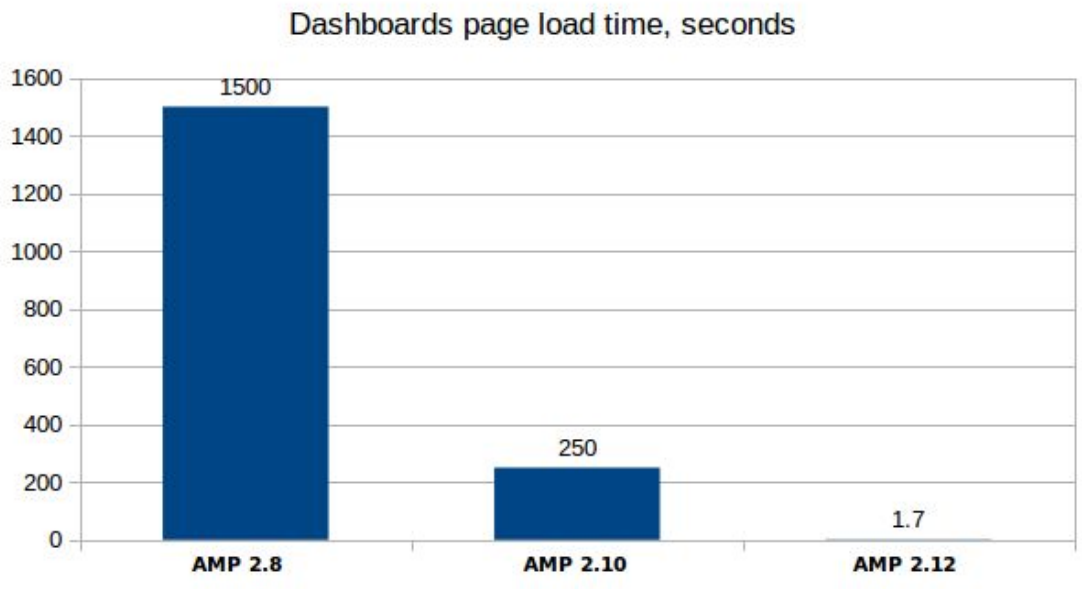

Figure 13 - Dashboards Page Load

The AMP maintenance has also improved in AMP 2.12, since MonetDB that was dedicated for Mondrian is no longer used, hence no maintenance and troubleshooting of MonetDB monitoring is needed anymore. Additionally the optimized AMP 2.12 Reports Engine has brought back AMP memory consumption to a simplified server configuration:

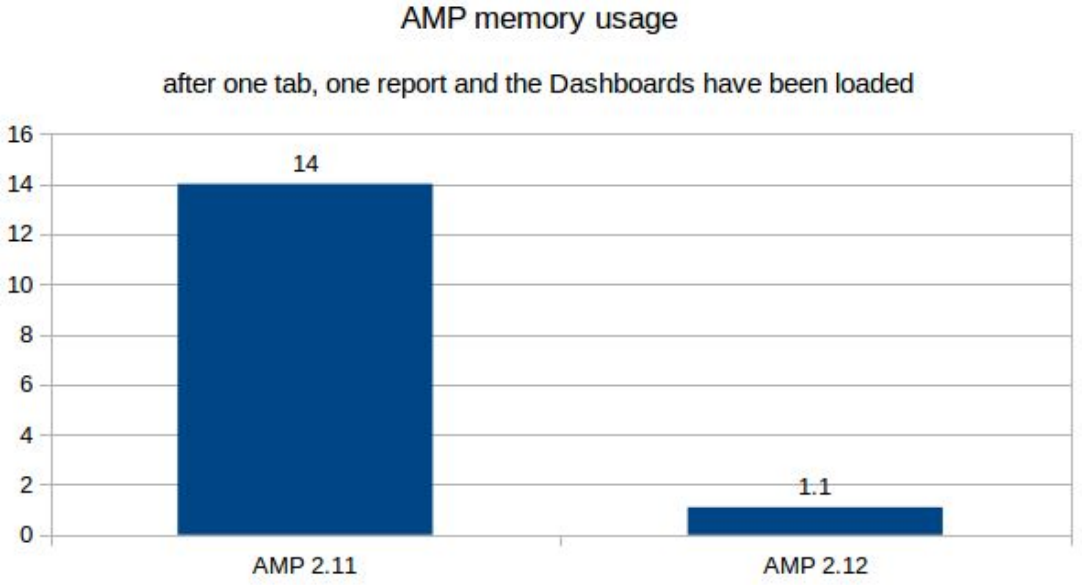

Figure 14 - Memory Usage