Technical Opportunities

Funchal | 13.12.2009

Contents

Executive Summary

Mambu is a vision for an online portfolio management software service for growing microcredit organizations. It outsources the technical challenges for these organizations and allows them to focus on their core business. It is affordable, designed for usability and allows the organization to access their information anytime, anywhere.

While there will be countless small technical challenges to overcome when building the Mambu service, we explored few interesting technical opportunities to some amount of detail over the course of the capstone project. Some of these opportunities are highly interrelated; some are just very brief discussions of issues captured during the design process. While this cannot be a full discussion of each of these opportunities we intended to capture them in the following chapters for future discussions.

While implementing the service from scratch will result in a clean and consistent codebase and look and feel of the interfaces, there is an open source solution available – Mifos – that could be adapted and allow a faster time-to-market for the Mambu service. The first chapter of this report discusses this opportunity and the risks and advantages involved in this – as we conclude – shortsighted path. Closely related to this conclusion is the topic of finding the right base-architecture for Mambu as a SaaS solution and how scaling the solution to a very large client base can be achieved through horizontal scaling.

In dialogues with MFI executives we experienced the issue MFIs might have in trusting their sensitive client data to an online service provider. While this risk – further described in the Business Viability Whitepaper – can be mitigated by having Promosoft as a traditional banking software solutions provider backing Mambu, we feel that transparency of the processes involved in building and maintaining Mambu is another opportunity and is briefly highlighted in the Living Transparency chapter.

In recent years frameworks and standards for interface creation have evolved and provide stable solutions for rapid interface development of large-scale web applications. We briefly compare two of these frameworks – Flex and GWT – in terms of their suitability for Mambu.

Another opportunity that has been explored as a part of the capstone project is the use of mobile technologies in the field, in particular digital pens and PDAs or Smartphones. We discuss the technical implications of building an infrastructure for these services in the final chapter of this whitepaper. While these technologies are an appealing solution for larger Microfinance organizations, they don't appear to fit to an initial release of Mambu that will be targeted specifically towards smaller ones.

Overall, we conclude that Mambu is a challenging but feasible undertaking. SAAS as a deployment model has grown into maturity and best practices are emerging which can be applied in the Mambu development. At the same time, services like Amazon EC2 are making cloud computing a real possibility for even smaller development teams allowing them to create scalable solutions at a reasonable cost. For Mambu to be successful it needs to be built from the ground-up and emphasize its strengths on a strong architecture with a unique and appropriate user interface interaction design on a solid framework. With the right team and solid system design, the tools are available to develop, deploy and pilot test a great product in a reasonable timeframe.

Mifos Adaptation

Mifos is a web-based management information system (MIS) supporting all key operations of a Microfinance Organization. Similar to a SaaS solution, it is a software package deployed on a web server inside or outside an organizations premise and accessed over local area network (LAN – if inside) or the internet (if outside) via web browser (such as Internet Explorer or Firefox). However, Mifos is designed to serve one organization per installed instance, which results in considerable amounts of hardware and software expenses: It requires a server operated and maintained by the organization using the system and big investments into networking infrastructure if it is to be used in more than one branch.

Mifos was successfully deployed for organizations of various sizes serving anything from 4.000 up to 300.000 clients. It was built by a professional software development team comprising developers of IBM and the Grameen Technology Center. It is therefore based upon a substantial amount of subject matter expertise rooted in the origins of the entire industry.

Given this setting anyone aiming to create a SaaS solution for the MFI industry would be reckless not to take a very close look at this system. It covers more functionality than any initial SaaS solution could provide and it represents a valuable resource of best practice knowledge that was built into the system.

The following sections highlight potential ways of adapting Mifos to serve multiple clients in a potential SaaS environment and how Promosoft could benefit from the best practices applied to the Mifos design.

Mifos Licensing

Mifos is licensed under the Apache 2.0 Open Source License[1] which allows for the reuse of Mifos' code, even in proprietary developments. The license grants rights to use modify and distribute the software in any environment, as long as it is distributed with appropriate note clearly stating the origins of the used Software and contains a copy of the License.

Software which makes use of source code licensed under the Apache 2.0 License does not have to be licensed under the same license. Also modifications on source code which is licensed under Apache 2.0 are not required to be sent back to the author of the original source code. However if software is published which is based on source code originally licensed under Apache 2.0 it has to retain all original copyright notices, clearly highlight the modifications that were made to original source code and contain a copy of the Apache 2.0 License itself.

This licensing setup leaves a lot of liberty to make use of the expertise that has been implemented in Mifos already. The following sections describe three different options that would leverage the existence of Mifos as a basis for a SaaS solution for microfinance.

Options for reusing Mifos

Based on the assumption that Promosoft is aiming to offer a scalable but lightweight solution to MFIs, Mifos could be reused in three different scenarios, all of which conclude in a product line dedicated for MFIs.

1. Virtualization: This is a take and run scenario. No major modifications to the core would be required and further developments embrace the upcoming releases provided by the Community.

[1] http://www.apache.org/licenses/LICENSE-2.0.html

2. Adaptation: This scenario includes heavy modifications to the Mifos core in order to transform it into a multi-tenancy solution that can be run on cloud-computing infrastructures.
3. Recreation: This scenario makes use of the knowledge built into Mifos but does not reuse any source code directly. A lightweight and easy to use SaaS solution is built newly from ground up.

Virtualization

In a virtualization scenario the provider would take Mifos as it is and create or buy[2] a platform which would allow the provider to rapidly deploy and manage multiple Mifos instances running on virtualized servers in a datacenter. All development efforts on the provider side would be dedicated to running Mifos in the virtualized environment, implementing or supporting the abstraction and backup layer required to deploy new instances on-demand. Additional effort will be required to establish billing and support infrastructures.

The interaction with the Mifos development community would be minimal in the beginning and rise over time as resources become free once the virtualization environment becomes stable. Over the long run substantial effort could be made to contribute to the evolution of the Mifos core in terms of developments on features on the roadmap or user experience enhancements to the core features.

Additional non-exclusive list of tradeoffs for this scenario are given in the following table:

Positives	Negatives
Very fast time to market due to maximum leverage of existing solution	No influence on the currently mediocre user-experience of Mifos
Comparatively cheap solution does not require heavy upfront investment	Updates of the software have to be deployed to many instances for an extra fee
Very close cooperation with the Mifos development trunk	Bound to Mifos release dates, no own scheduling ability for enhancements
Many features in an initial deployment	Scaling problems once many customers adopt the solution

Adaptation

In an adaptation scenario the provider would partner with a specialized service provider – so called Software as a Service Enabler[3] – in order to adapt the Mifos core towards being able to serve multiple tenants. Substantial capital investment will be required to finance the adaptation efforts, but pay off on the long run due to the buy-in SaaS expertise of the external provider. The outcome can be expected to be a solid starting point for further developments with infrastructure for signup, billing and support already in place.

Additional non-exclusive list of tradeoffs for the adaptation scenario are given in the following table:

[2] See available solutions at: http://www-03.ibm.com/systems/virtualization/infrastructure/
[3] E.g. http://www.saas-attack.com

Positives	Negatives
Fast time to market due to buy-in expertise and reasonable leverage of existing solution	Minimal acquisition of expertise in venture's core technology
Independence from Mifos community process and freedom to advance at own pace	Improvements in user experience possible only after branching of from Mifos core
	Initial solution will include a lot of features that are not specifically targeted towards small MFIs and have to be maintained over time
	Dependence on external provider during critical start-up phase

Recreation

In a recreation scenario many of the use cases that already exist in Mifos can be borrowed and refined in user tests. This scenario requires the heaviest upfront investment for a rigorous design and architecture phase, with the payoff of maximized control over the later codebase and a product targeted very specifically to the needs of an initial niche market. Efforts for creating an infrastructure for signup, billing and support can be reduced by employing a pre-made SaaS infrastructure solution[4]. Efforts for developments have to be reduced to a minimum set of features that create most value to potential customers in the targeted niche market and can later be extended to target a broader spectrum of customers.

Additional non-exclusive list of tradeoffs for the recreation scenario is given in the following table:

Positives	Negatives
A product very specifically targeted towards the niche market that is to be served initially	Extended time to market due to development time required for a shippable alpha release
Independence from Mifos community process but based on Mifos knowledge and best-practices	Major capital investment required for an extended period of development time
Excellent user experience through the application of user-centered design methods	Dependency on hiring experts for SaaS development and deployment can delay launch

Conclusion

Both the virtualization and the recreation options are very appealing, however the pure adaptation option does seem to come with very expensive tradeoff costs. The appeal of the virtualization option mostly draws from its close ties to the Mifos core development group. Over time a development team could be dedicated towards advancing Mifos as an Open Source solution and make a significant impact on the community process. Transitioning from the SaaS solution to an on-premise Mifos setup once the MFI has grown to a reasonable size would be enabled through a simple SQL dump option.

The appeal of recreating a solution from scratch comes from its superior user experience. One of Mambu's strongest differentiators over time would have to be a superb user experience, and this must be done from the roots of the applications design; not just at a skin-deep surface level. A truly lightweight solution for small MFIs is missing in the marketplace right now and the proposed SaaS solution design would close this gap. The biggest problem with any of the options based on Mifos however is that Mifos has not been build

[4] E.g. see options available on http://www.apprenda.com

to scale horizontally and does not support stateless interactions, both of which will be explained the following chapters.

Tiering and Choosing a SaaS Level

In a Software as a Service scenario there is multiple fundamental architectural decisions to take. One of the most impacting ones is that of choosing the right SaaS "Maturity Level" for the desired application. This decision has implications on the business model, payment structure and all levels of architecture that will be implemented in the service.

Single Tier – Single Tennant

Figure 1 visualizes how Promosoft's business model works today, using the Banka product as an example. The software is developed and updated at Promosoft premises and sold or licensed to multiple clients. Clients have to buy and maintain the server infrastructure to run the software product on their premises. Customization – parameterization in the case of Banka – is performed by consultants in close collaboration with the client.

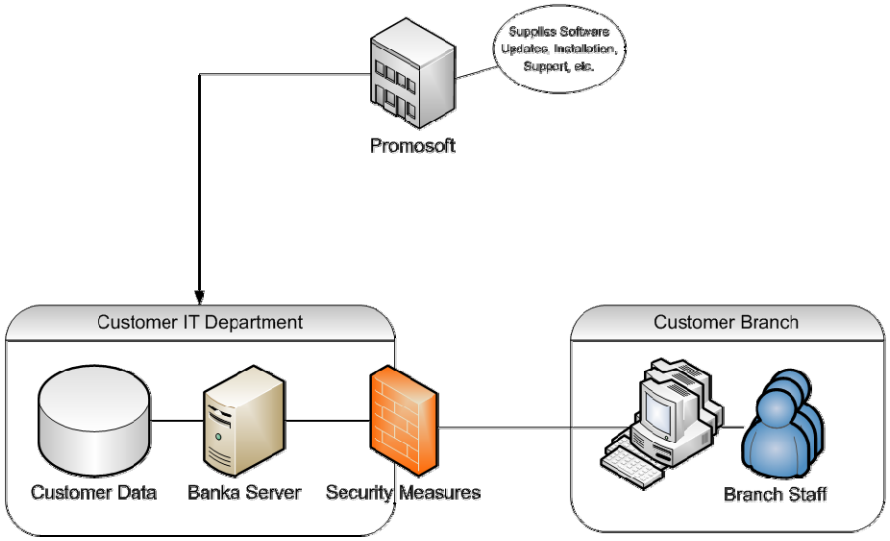


Figure 1: Software as a Product Model

Most business to business software products follow this mode of deployment and the largest portion of these implement a single-tier architecture, that is: All parts of the application, database, business logic and interface elements are deployed on one server, at times with redundancy and backup systems in place. Once the server runs out of capacity or lacks performance as client numbers are increasing, it is replaced by a stronger machine (see also Horizontal Scaling chapter).

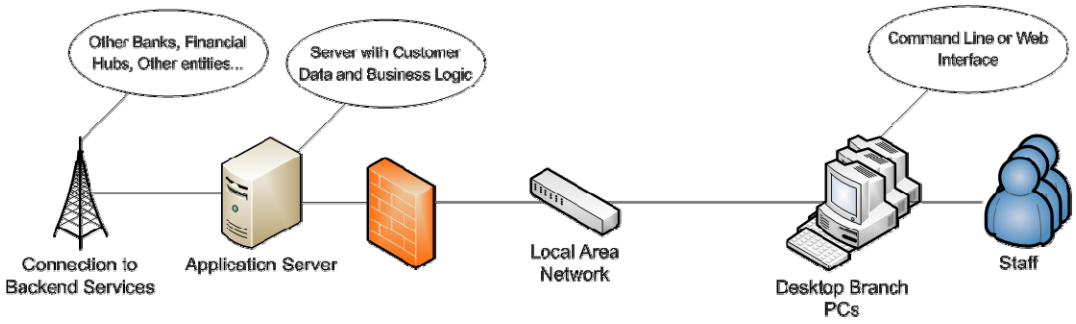


Figure 2: Single Tier On-Premise Architecture

Figure 2 shows this setup including connected client and backend systems. While there could be multiple tiers involved on the server machine itself (e.g. with database abstraction and business logic layers involved) this product-licensing model can hardly be transformed into an efficient SaaS scenario. Application Service Provisioning (ASP) is one option where the hosting of a product can be outsourced to an application service provider. Each client can be served with a virtualized server instance in this scenario and the pricing is adjusted based on the capacity of the virtualized server instance.

Nevertheless, the above described mode of deployment is still only fully feasible for large scale deployments as prevalent in the banking domain and close collaboration between software product provider and client is necessary to set up the various banking products and customized modules. However it has severe limitations when it comes to applying it to the largest portion of customers in the microfinance domain, the small and medium sized MFIs.[5]

Introducing SaaS Maturity Levels

To implement a successful SaaS solution one must look at three fundamental requirements. These includes scalability to multiple clients, multi-tenancy to control internal costs and customization thgouh configuration. These issues are address by four basic levels of SaaS Maturity levels:

Level 1 is an ad hoc or custom service where nothing is done for allowing multiple tenants to run on the same instance. Each customer has a different copy of the software.

Level 2 is a configurable service where the same software is tailored for each client via configuration but all use the same code base. However, computing power is now shared amongst instance.

Level 3 is a configurable multi-tenant solution. The UI can be customizable per tenant as can the business rules and data models.

Level 4 is a fully scalable, configurable multi-tenant solution and is described here as the best and most appropriate for Mambu.

Multi-Tier – Multi-Tennant

The prevalent deployment model in SaaS involves the installation of one single codebase on a scalable server infrastructure, that is maintained, updated and supported by the solution provider. Figure 3 illustrates this

[5] See Business Viability Whitepaper “Microfinance Software Market”

model for the Mambu service. In an ideal scenario Promosoft or the respective service provider would outsource the hosting of the Mambu service to a cloud computing service provider such as Teremark's vCloud Express or Amazon's EC2 service. The service is managed over the Internet by Promosoft and many clients from all over the world access it. All their data is stored and managed in the same server infrastructure.

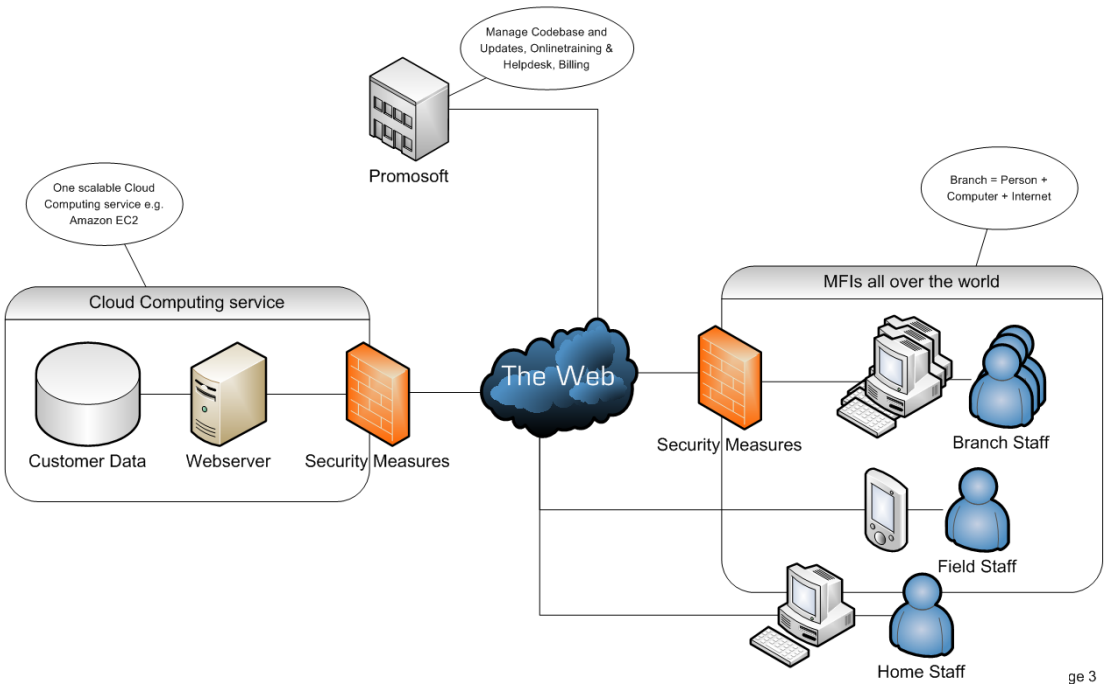


Figure 3: Level 4, Multi-Tennant-Efficient

This setup comes with an immediate global scalability (see chapter "Horizontal Scaling") and therefore allows a business model that is cost effective for small clients. As there is only one codebase to maintain, all the clients will get all of the updates at the same time, as a part of their subscription contract. Observably. it comes with limited customization abilities as data models have to be consistent to large extends throughout the client base.

Behind the scenes of a scalable SaaS architecture — like the one Salesforce.com employs — there are multiple tiers involved that can be scaled independently (see Figure 4). Separating at least the data from the business logic tier offers the great benefit of being able to have fine grained adjustments to either storage or processing/performance requirements and therefore offers more cost effective use of capacities. Having an additional front-end tier would allow to optimize performance for serving different platforms. Ultimately providing dedicated servers for each web browser requests, mobile application requests and third-party web service connections will allow optimize the performance of data communications towards these different access mechanisms.

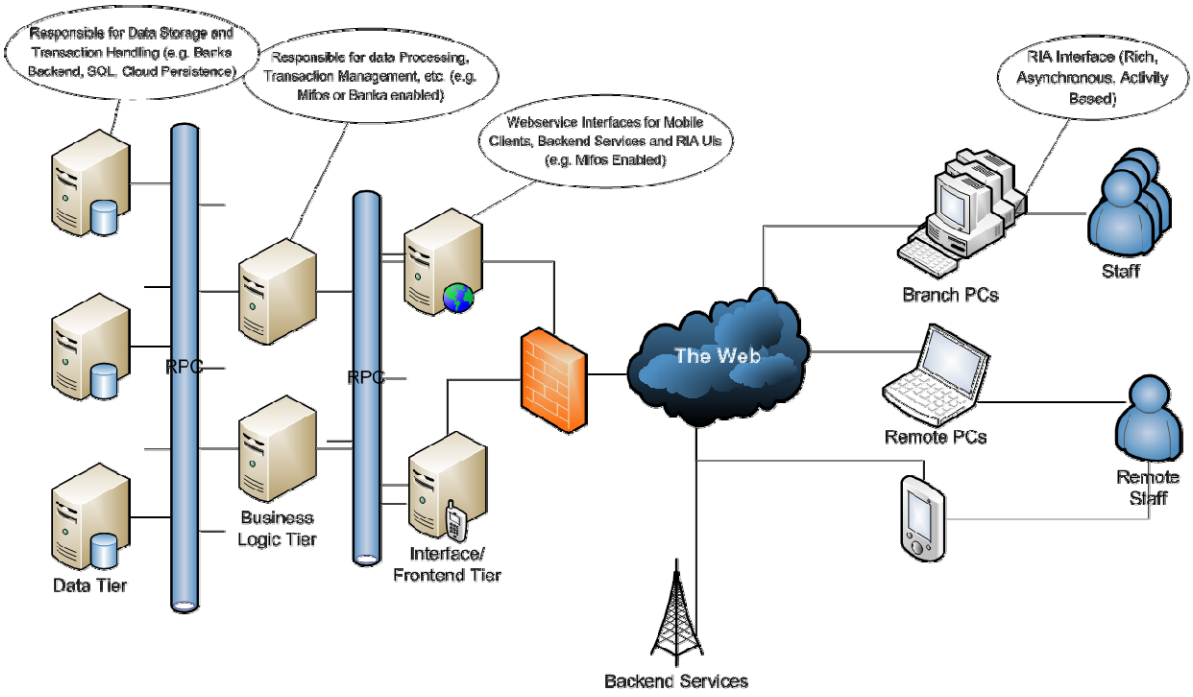


Figure 4: 3-Tier Architecture

The idealized Level 4 model presented above works perfectly in domains with little government regulation, such as in the CRM field as Salesforce.com introduced it. However since the Microfinance sector is heavily regulated in some countries and less regulated in others, there could be limitations to the Level 4 model in some markets. One objection the regulating central bank of a country might have is that financial client data has to be stored on servers within the regulated country itself. While our field research did not reveal concerns about this issue in Mozambique there might be other countries where a problem like this arises, especially in more organized markets with higher MFI penetration rates. In this case the scenario becomes slightly more challenging as now, while still having to provide a scalable architecture for multiple tenants at a competitive pricing model, the same scalable infrastructure as described in the above Level 4 scenario has to be cloned for different countries. Depending on how many countries impose requirements as this it is unlikely the service can be rolled out to these countries without partnering with local IT service providers who will take care of the datacenter operation and billing part (see Figure 5).

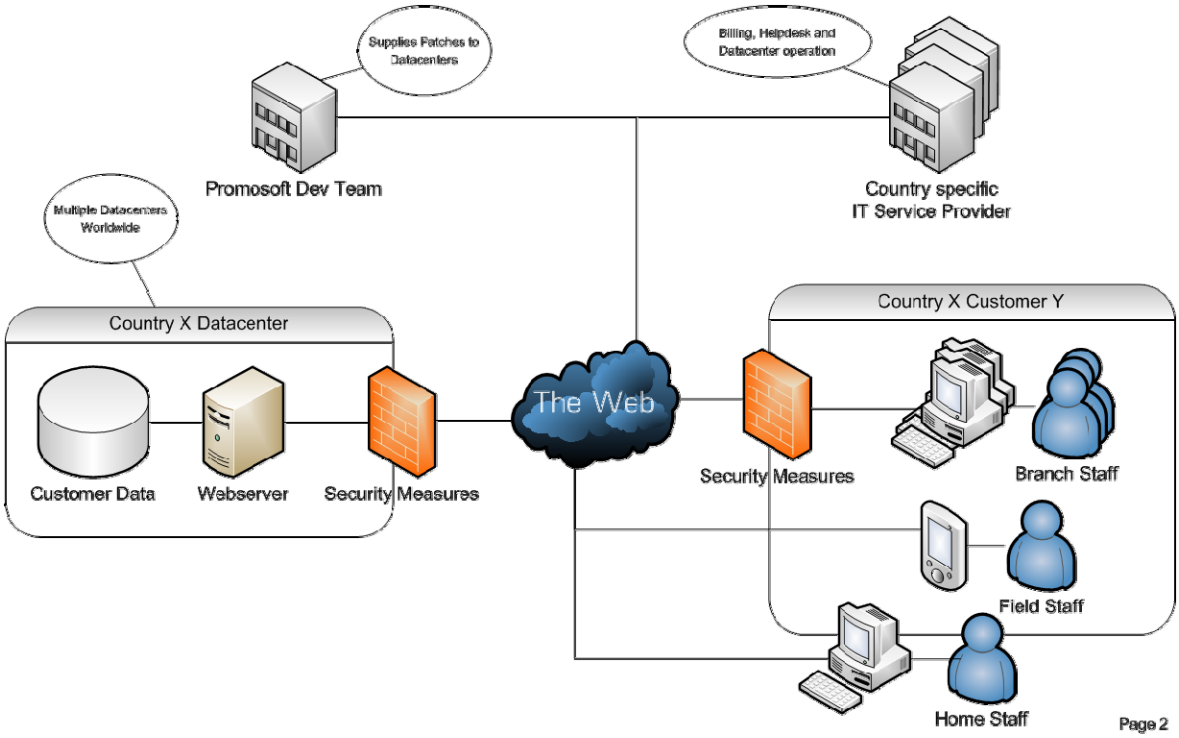


Figure 5: Level 3, Multi-Tennant-Distributed

Scaling-out with Cloud Computing

Just because a system is blindingly fast for 1,000 users and 1 GB of data, it's not scalable unless it can maintain that speed for 10 times the data with 10 times as many users. – Cal Henderson, Former Chief Architect at Flickr

Performance and scalability are two fundamentally different issues. While a system can be tremendously high performing in small scale, it doesn't necessarily mean that it will be able to perform as well once datasets and user counts increase. On the other hand, the ability to scale usually comes with the ability to perform better by scaling-out. The following briefly explains how building a scalable online service involves fundamental architectural decisions and how these should be considered when building Mambu.

Designing a scalable system is by no means a decision for or against a particular technology. While Java Enterprise is oftentimes used synonymously with scalability there can be PHP or Ruby on Rails web applications that are just as scalable and high-performing. In fact Java indeed comes with the risk of building heavy, stateful, systems that visualize the difference between horizontal and vertical scaling.

Vertical Scaling

Many first generation web applications have been built in a stateful manner: User profiles and customer data were stored in sessions in the memory of big server machines, optimized for performance while the respective user is logged in to the system. Systems which were architected like this are most likely prone to vertical scaling, which simply means: once a server machine runs out of capacity, it gets replaced by a more powerful server machine. This process goes on and on as the user base grows, in some cases having staging and replication of servers involved. It is really easy to design for vertical scaling as software can be developed on a local machine and after deploying it to a server there is little interaction necessary. However the problem with this is that the costs for faster servers don't scale linear with the number of users in the system. When building a system that is open to a potentially unlimited amount of users or where the size of the user base cannot be foreseen, as in Mambu, the cost model of vertical scaling just doesn't make sense.

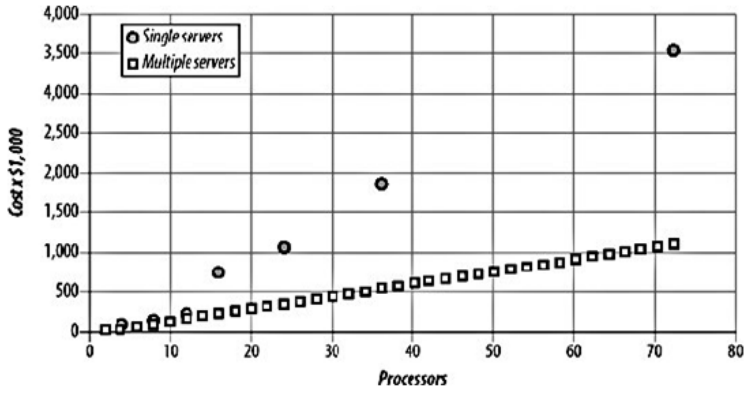


Figure 6: Horizontal scaling vs. vertical scaling

Horizontal Scaling

Scaling a system horizontally is the same as vertical scaling in that once a capacity limit is reached, new hardware is added. However in this scenario the application is not transferred from one machine to a stronger one, but instead an additional low-cost machine is switched on that provides the required CPU and storage capacities. Especially in SaaS model this allows for very fine grained adaption of capacity as no major investments into new server hardware are required. As capacity needs are directly correlated with the

number of paying accounts in the system, costs for the server infrastructure scale linear with the revenue from pays accounts.

Managing a lot of server instances is tedious and an expensive administrative task. It involves selecting the appropriate hardware for both processing and storage needs, replacing old hardware, installing operation systems and management software. Very recently, however, cloud-computing service providers have started to provide exactly these services. They make it increasingly easier for software service providers to outsource this administrative burden to some entity that is capable of doing it at a much bigger scale to a fraction of the cost.

Amazon Web Services is the biggest provider of cloud hosting solution in the market today. It provides the entire infrastructure that software service providers need. This includes server instances in multiple sizes, database server instances, load balancing, redundancy, backup and worldwide content distribution networks. With Amazon EC2 scaling-out for new capacity can be reached within a couple of minutes. Image files for the server instances are prepared by the service provider, made available in EC2 and can be switched on and off on-demand. Updates to the service software will in this case be performed by updating the server image file and relaunching the active instances with the new image.

Table 1 shows the prices of the Electronic Cloud Compute (EC2) products offered by Amazon. It also emphasizes how prices for larger server machines rise exponentially while adding single small instances can be performed for a very competitive price.

Server Instance Performance	Cost per Hour	Cost per Day	Cost per Month	Cost per Year	Number of Small Instances for Price
Small	$ 0,10	$ 2,28	$ 68,40	$ 832,20	1
Large	$ 0,38	$ 9,12	$ 273,60	$ 3.328,80	4
Extra Large	$ 0,76	$ 18,24	$ 547,20	$ 6.657,60	8
Double Extra Large	$ 1,34	$ 32,16	$ 964,80	$ 11.738,40	14
Quadruple Extra Large	$ 2,68	$ 64,32	$ 1.929,60	$ 23.476,80	28

Table 1: Example costs and scale for Amazon Elastic Cloud Compute instances (Prices as of Dec. 2009)

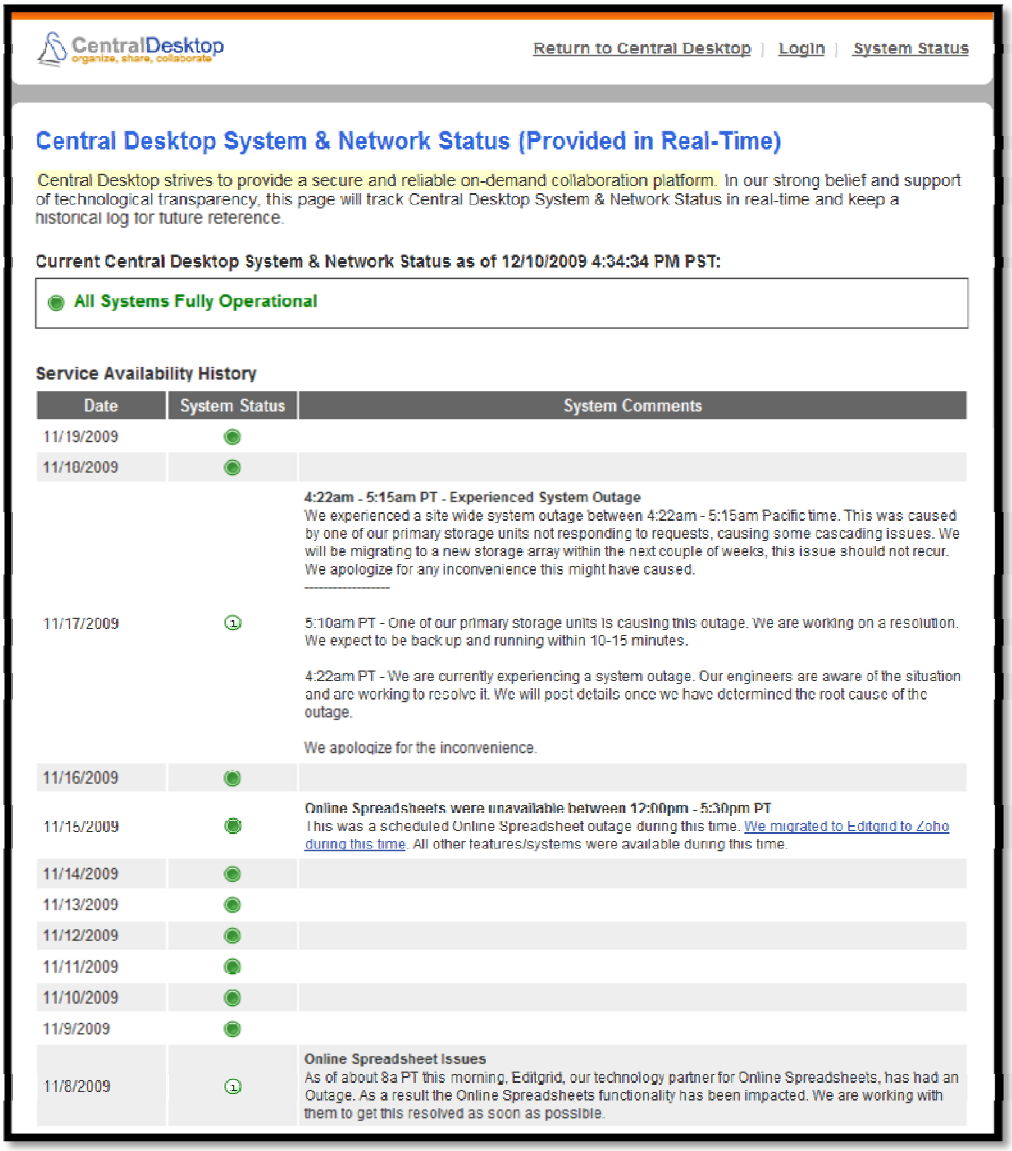
Design Implications

Creating a system to be horizontally scalable has to a decision from the very beginning as it has design implications on all levels of the system architecture. The system has to support a design that Google Developers refer to as “statelessness”, which is also used in Enterprise Java Beans. In a stateless system the client – in case of Mambu a web browser – does not require to communicate with the same server again and again. Ideally each chunk of data that gets transmitted from the client to the server can be associated with the client’s account just by looking at the data itself. In that case none of the servers will have to maintain a “state” for clients such as heavy sessions that include data about the client’s operations. Each request to the server can be authorized, processed and answered by any one of the server instances that has enough capacity to answer it. In a less ideal scenario, client databases can be cut to pieces on a per-country level and only server instances within this given country can serve client requests to provide this data.

Living Transparency

There is no software that is free of bugs. Truth is hard to find, but this one has been confirmed over all the digital years by countless organizations and development teams. All software has human origins and so does it absorb errors, imperfections and lacks of completeness and logic which emerge at the time of its creation. While development techniques like code reviews, continuous multi level testing of code units, modules and data integrity and the continuous integration of enhancements into production code can prevent the most severe failures of a system they cannot prevent all. In a system like Mambu with potentially thousands of tenants relying on its stability it is critical to communicate both the above mentioned techniques to prevent failures and also, but probably even more important, communicate once something has failed.

Taking Central Desktop as a SaaS provider for project management and team collaboration is a nice example of how the open communication can be achieved. Instead of hiding their weak spots from the general public, they have chosen to publish whenever a failure has occurred and which steps are being taken or have been taken to resolve the issue and make sure it will not reoccur in the future.



CentralDesktop
organize, share, collaborate

Return to Central Desktop | Login | System Status

Central Desktop System & Network Status (Provided in Real-Time)

Central Desktop strives to provide a secure and reliable on-demand collaboration platform. In our strong belief and support of technological transparency, this page will track Central Desktop System & Network Status in real-time and keep a historical log for future reference.

Current Central Desktop System & Network Status as of 12/10/2009 4:34:34 PM PST:

All Systems Fully Operational

Service Availability History

Date	System Status	System Comments
11/19/2009		
11/10/2009		
11/17/2009		**4:22am - 5:15am PT - Experienced System Outage** We experienced a site wide system outage between 4:22am - 5:15am Pacific time. This was caused by one of our primary storage units not responding to requests, causing some cascading issues. We will be migrating to a new storage array within the next couple of weeks, this issue should not recur. We apologize for any inconvenience this might have caused. --- 5:10am PT - One of our primary storage units is causing this outage. We are working on a resolution. We expect to be back up and running within 10-15 minutes. 4:22am PT - We are currently experiencing a system outage. Our engineers are aware of the situation and are working to resolve it. We will post details once we have determined the root cause of the outage. We apologize for the inconvenience.
11/16/2009		
11/15/2009		**Online Spreadsheets were unavailable between 12:00pm - 5:30pm PT** This was a scheduled Online Spreadsheet outage during this time. We migrated to Editgrid to Zoho during this time. All other features/systems were available during this time.
11/14/2009		
11/13/2009		
11/12/2009		
11/11/2009		
11/10/2009		
11/9/2009		
11/8/2009		**Online Spreadsheet Issues** As of about 8a PT this morning, Editgrid, our technology partner for Online Spreadsheets, has had an Outage. As a result the Online Spreadsheets functionality has been impacted. We are working with them to get this resolved as soon as possible.

Figure 7: Central Desktop "network status" page with all errors that recently occurred

Figure 7 displays their “System & Network Status” page which is accessible from any other page in the web application over a link on the footer of each page. In case something out of the ordinary happens, customers can check this page and see whether the service provider is already aware of the problem, file a support request or double-check whether it could be a problem on his own side.

Certainly in a system like Mambu, which is used to manage the core businesses of it’s clients, designing for and reaching a close-to 100% uptime is critical. Nevertheless, if parts of the system fail or are inaccessible for a short period of time, publishing this information will give the service provider credibility and ultimately make customers more confident in using the service. As customers can then immediately see how the technical team of the service provider works to solve their issues, they will feel their continuous investment in the service is actually used to enhance the service apart from the frequent updates to the system.

The decision to publish information becomes more complicated when talking about bugs or support requests. While open source software providers can easily maintain online forums like Google Groups to discuss support issues, where clients can help out other clients with their problems in using the service, this becomes problematic in a software service that involves handling money and sensitive client information. Publishing bugs online has its limitations for exactly the same reasons but has to be treated more carefully as it might include even more sensitive information about system deficiencies that could be exploited by malicious users. However, having an open discussion forum where clients can talk to each other, request features and event vent some inevitable frustrations can be beneficial.

Embracing Standards and Frameworks

Interface frameworks have recently gained a lot of attention in the web development community. There are numerous framework that facilitate the development of AJAX based web applications and two major frameworks which could be used in the creation of the Mambu service are briefly compared in this chapter

Google Web toolkit is an interface development framework for web applications, it is open source, supported, initiated and used by Google. It allows developers to write interface code in the Java language and later compiles it to JavaScript versions that can be run in all major browsers. There are further additions available to GWT, such as SmartGWT and ExtGWT, which further extend the available widget library and allow development for a better user experience. Flex on the other hand is a proprietary solution provided and maintained by Adobe and used as a basis for YouTube and other experience-rich applications. Flex interfaces are compiled to Adobe Flash content and require the installation of the Flash plug-in on the client’s web browser.

While Flash plug-in penetration is fairly high in a private setting, it is well known that many corporations restrict or prohibit the installation of Flash on their computers. One would have to perform further studies as to how far Flash is distributed in the Microfinance domain in order to conclude whether this would be a constraint for the deployment of Mambu. However we feel that GWT has a major advantage here. It is truly cross-browser compliant and platform compatible and can be used on Linux, Mac, Windows, Android, iPhone, etc, whereas Flash will most likely never be. As GWT fully integrates with the document object model (DOM) of a website, everything else displayed on any particular webpage that involves GWT elements can be developed far easier than in a fully fledged Flex scenario.

Both frameworks are highly optimized in terms of their bandwidth usage as all interface elements are transferred only once and cached on the client side from then on. As only business logic data is transferred

over the network from then, both solutions provide a very fast and responsive user experience. Flex has still an advantage over GWT in terms of user experience. Designers can build the user interface and it can be added and changed without much effort, as Flex is a development environment originated in the design-domain. GWT and its extensions provide a wide, but limited set of widgets in their original libraries and it requires significantly higher efforts to develop applications to the same level of user experience. New widgets will have to developed and maintained in Java code, CSS and HTML.

Talking about future versions of both frameworks there is no clearly visible advantages in either of the frameworks. GWT underlies a community process and decisions for further developments are made in an open-source style. Adobe is strongly committed to Flash and Flex and will keep supporting it for a long time. Even though there are standard conform solutions for animations and rich user experiences on the way with HTML 5, Flash will stay the No. 1 for this in the marketplace for any future it is reasonable thinking of. Also both solutions are easily unit testable and conform to common development practices, so they would both fit into the Mambu development lifecycle.

Overall we feel that GWT in combination with Ext would be a very suitable solution to be used in Mambu, which apart from the points mentioned above mostly comes from the familiarity of most developers with the Java programming language. Learning GWTs particularities can be achieved comparatively fast, while capacity building for Flex appears to be tedious and long-term benefits of using either of both are not readily apparent. Another strong argument for the usage of GWT over Flex in the Mambu solution is the upcoming standard of HTML 5. HTML 5 will have built-in support for offline caching of data which is highly relevant for the Mambu scenario, as internet availability could be a challenge to overcome in certain regions of the developing nations Mambu would be targeting.

Using Smartphones and Digital Pens with Mambu

Two mobile technology options were explored as an extension to the Mambu solution and their feasibility for implementation is discussed here. The expression “Smartphone” is used as a cover-all category for mobile devices including PDAs, web-enabled phones and touch-screen interface phones. The Digital Pen solution refers specifically to using a digital paper and pen solution such as those developed and licensed by Anoto. Devices such as Netbooks and Tablet PC are not considered as a separate mobile solution since they would be able to function and be developed for like any other PC platform.

Smartphone Implications

Smartphones are the simplest to develop for, the cheapest to maintain and the most flexible in field operations, even though it would have to be tested which activities of the Mambu service would make sense to perform on a smartphone. From a technical point of view, developing a mobile version to access the service with limited interface interactions would be simplest and enable to reach the most devices. Assuming the Mambu service is cleanly implemented in a multi-tier architecture, extending it to a usable mobile interface is not a greatly challenging task and could make use of existing web service interfaces. It would certainly be beneficial to accomplish this early on at the very least as a proof of concept of the anytime/anywhere accessibility of Mambu and to verify response times over mobile internet connections. This certainly speaks for the implementation of a clean multi-tier architecture and strong interface to backend decoupling which further minimizes data transfer overhead for interface elements.

A Smartphone-optimized web interface for Mambu would allow it to be accessible by most phones but for optimal ease of use, it should be catered to different screens. The strongest benefit would come from having a dedicated application in the smartphone which would be a client of Mambu. This would allow it to be used and accessed regardless of connectivity and to be synchronized with the main server at a later time. The Symbian platform would be the most logical to develop for initially, with the OS currently accounting for more than 50% of smartphones including a large percentage of the budget smartphone market. However, the Symbian platform is notoriously difficult to develop for and would require a unique skillset of expertise on the team. A looming alternative would be the Android OS. Quickly gaining market share, it also offers the clear advantage of being available on cheaper smartphones in the near future. With multiple manufacturers joining the open-handset-alliance powering the Android OS, prices for lower-end version of these phone will quickly drop over the coming years. The Android OS, as a web enabled system, provides native support for web service connectivity and would therefore be an easy and natural extension to develop on.

As a whole, if smartphone capablities are taken into account from the beginning of the design of the service, they have the potential to offer rich interaction with low architectural overhead. They would be suitable to MFIs who are willing to risk having a moderately expensive device with their loan officers. As field research revealed, not all MFIs trust their loan officers to carry costly equipment – as they could try to sell it off and later claim they “lost” it – the integration of smartphones into their workflow would require some training and capacity building.

Digital Pen Implications

A digital pen – combined with specially prepared paper forms – offers very compelling options from the point of view of an MFI using it. However, the architectural overhead and costs associated with its implementation may make it prohibitive to all but the best-off of the MFIs, which is in clear conflict with the market that Mambu is targeting.

A digital pen would allow an MFI to print evaluation sheets through Mambu and have the data sent back the filled information either through Bluetooth and a phone, or by synchronizing with any PC with an internet connection. The clear advantage of this is how seamlessly it appears to integrate with the workflow. Credit officers would simply continue to work as before with the one additional step of having to synchronize their pens. However, the back-end overhead of this solution is quite considerable.

For starters, all evaluation forms would need to be available in the special Anoto format. Since each form is unique, this requires the generation and printing of these forms. This not only costs a small fee per form, but also has the overhead of having to transfer very large files (a few megabytes each) for each form. This may not be feasible for many organizations. Even having obtained the forms, the data sent by the digital pen still need to be verified to be translated to digital information accurately, a process would need to be introduced into the MFI and into Mambu to deal with process and the special training required on MFI side. Additional costs compound from a monthly fee per pen.

From the development point of view, implementing architecture to support the digital pen would be a large overhead and require major adoptions of the service. As well, a new billing model may be necessary to be able to support MFIs using the digital pen solution so as to allow Mambu to recuperate its additional costs for providing the digital pen services. For the MFI using the pen, a lot of complexity would be abstracted by Mambu, but the pens are still an expensive solution and one that would cause difficulties in maintaining as expertise in the pen hardware is rare. Alternatives to the Anoto platform and licensing may become available in the future which may make the digital pen option more attractive and affordable.

Conclusions

Overall, a smartphone solution is cheaper to deploy and cheaper to operate and a well-designed interface can reduce the complexity challenges of the device for it's users. It also offers a clear advantage over a digital pen in that two-way communication is possible, whereas the pen is a write-only device.