Scaling ERPNext

Leveraging various techniques to scale ERPNext to your business needs

WHITE PAPER | MARCH 2020

Scaling ERPNext

Leveraging various techniques to scale ERPNext to your business needs

Executive Summary

This paper covers the common challenges that users are faced with when running ERPNext at a massive scale. We explore strategies that can be employed to overcome such challenges. ERPNext is built on a three-tier architecture with a database backend, an application layer built on a front end. Apart from this, in deployments at large volumes, data will be fed by API integrations with payment and IOT platforms while the users will be consumers of data. In this paper, we cover scaling each tier and handling specific bottlenecks that appear with the growth of an ERPNext implementation. For large volumes of accounting or stock transactions, the “ledger” will become the bottleneck and this paper will also discuss strategies for partitioning the ledger.

Background

ERPNext is a 100% open source (Free Software) ERP product with full feature functionality licenced under the GNU General Public License. It has been in development since 2008 and has undergone several releases and is in production in more than 5,000 companies of different scales worldwide. Apart from Core Financials, ERPNext covers a wide range of business functions including Inventory and Warehouse management, Customer Relationship Management (CRM), Human Resource Management (HRM), Sales and Purchase, Project Management, Supplier Management, Helpdesk/Support, and much more.

ERPNext has been used in many industries and also has industry specific modules such as Manufacturing, Distribution, Retail Point of Sale, Education management, Healthcare management, Agriculture management.

With such functionality in seamless cohesion, it is natural to use the product for all the features and domains it has. Once a company implements ERPNext, the database is bound to increase with the growth of a company. Sometimes the hardware resources fall short, sometimes the database does, or in other cases, both.

Let’s take an example of a company named Vedmata which faced issues when scaling ERPNext. Vedmata is a non-profit organization which provides for the less fortunate and also sells other items to support their cause. Their most busy section is their POS specifically in the morning where it sees thousands of transactions across 70 centers. After reaching a large number of transactions, Vedmata faced challenges in performing simple transactions and viewing the ledgers.

The bottlenecks became so severe that submitting one item in POS took 20 seconds!

1. ERPNext Architecture

To understand how scaling is achieved in ERPNext, let's get an overview of the architecture. ERPNext has a three tier architecture with a database backend, a Python based application server and a JavaScript / web frontend. It is based on **Frappe Framework**, a full stack web framework built on Python and created by Frappe.

1.1 Frappe Framework

Frappe is a full stack web framework that manages the application including:

1. Metadata
2. Database operations (CRUD)
3. Controllers
4. Web Workers and Job Workers
5. Views:
    a. Admin UI (Desk)
    b. Portal UI
    c. Reports and Prints
6. REST API
7. Communication (Email/SMS etc)

1.2 Services

Frappe Framework uses the following services for running the application:

1. MariaDB (database)
2. Frappe Framework (Python) (web framework)
3. Redis (cache + queue)
4. NodeJS (socketio)
5. File Server

6. NGINX (Web Proxy)
7. Supervisord + Gunicorn (process manager)

1.3 Frappe Bench

The deployment is handled by Bench, a command line utility that helps install, manage, and run all the services required by Frappe Framework. It also manages site migrations. Bench is also a multi-tenant system where each tenant has its own database and files that are served from separate folders called **sites**

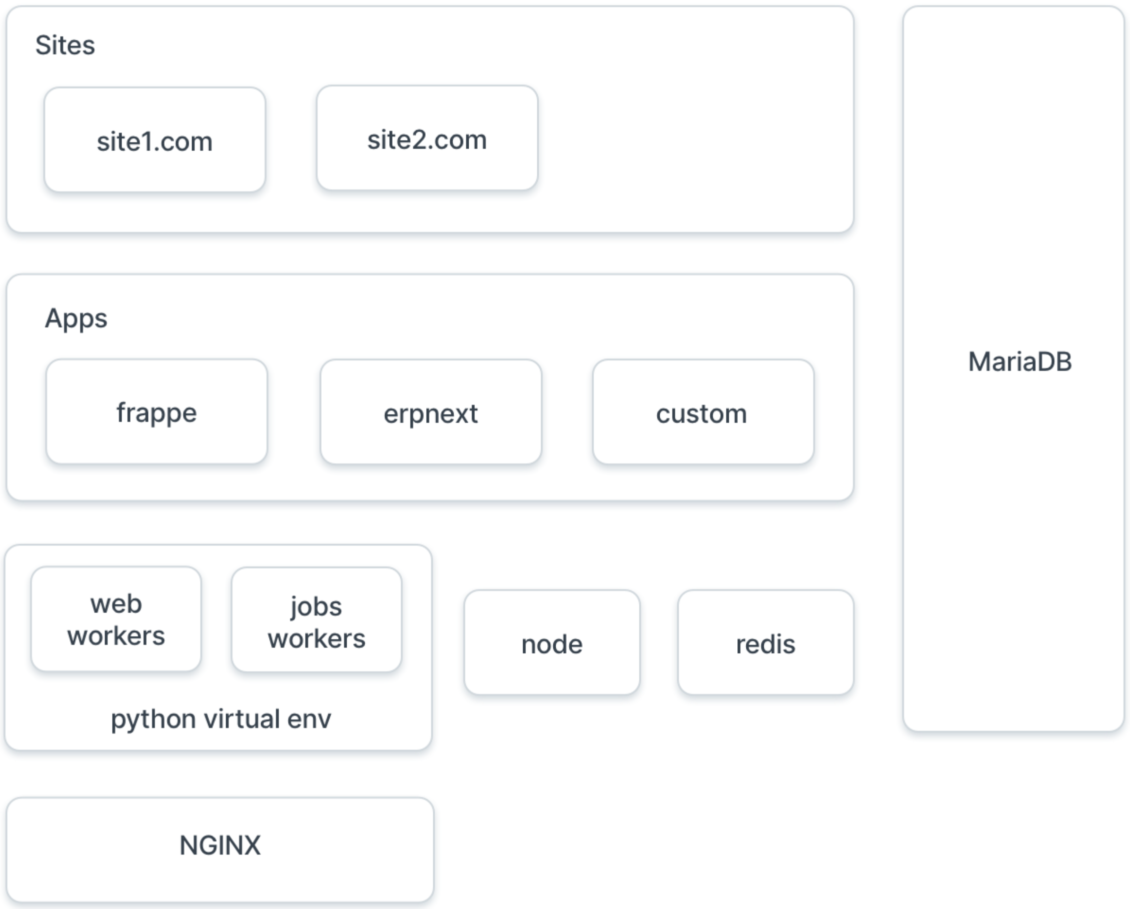

Diagram 1: Frappe Framework Components

2. Scaling Resources

In this section, we will discuss strategies to scale the services on which ERPNext depends. Each of these services have scalability strategies of their own. This effect multiplies and allows addressing individual issues in each component.

2.1 Service Oriented Architecture (SOA)

Each of the core ERPNext services can be moved to separate machines to implement a service oriented architecture. This way, we can independently scale the bottleneck services and also add machines to dynamically manage loads during peak times. Examples include end-of-month calculations like Payroll and peak hour POS transactions across different outlets of a single organization. The ERPNext service oriented architecture includes creating an ERPNext Cluster.

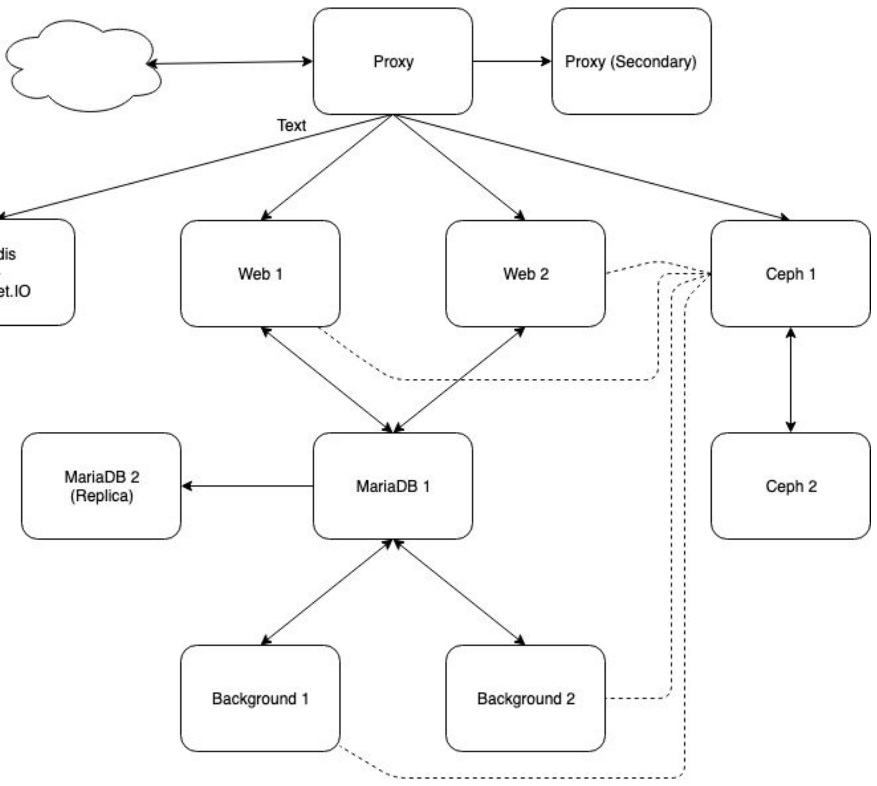

Diagram 2: ERPNext Cluster

The **ERPNext Cluster** is a set of machines that will create a service oriented ERPNext deployment. These can be deployed as Virtual Machines on a Cloud using an orchestration framework like Kubernetes or something similar.

All of the services in the cluster are independently and dynamically scalable.

2.2 Hardware Scaling

In Vedmata, hardware had become a bottleneck due to the large number of transactions. This was mainly due to 70 account users working simultaneously from different locations. Their 4CPU server was upgraded to a 12CPU server and RAM was increased to match the specs. Doing so decreased their server's load average from **8 to 2**.

ERP is both a memory and processing power intensive application, so a deployment with high scale must have sufficient RAM and CPU.

1. **RAM**: With large volumes of data, getting fast access to the data and the ability to manipulate it becomes critical.
2. **CPU**: If there are a high number of concurrent users or processes, then providing multi-core or multiple machines to all the parallel processes will become a necessity. Also for generating summary reports, it is important to provide sufficient processing power to ERPNext.

2.3 Database Scaling

In Vedmata, lots of database tables had major bottlenecks. There were 100,000 Sales Invoices and growing every day. This also meant there were about 300,000 entries in the General Ledger. Performing database scaling and optimization resulted in a dramatic reduction in Sales Invoice submission time. Due to bottlenecks, the submission time had become 10 seconds, and now it is under 2 seconds. We added indexing and caching to their databases.

Since ERPNext is a data intensive application, the first bottleneck usually is seen on the database service. Running and maintaining databases has been done by administrators for many years and there are multiple ways to optimize and scale databases.

2.3.1 Configuration

The first thing is to ensure that the database is configured to use the right amount of memory, page size, and cache. Since MariaDB uses the InnoDB engine, it is optimum to allocate the maximum amount of RAM to database operations and the InnoDB buffer.

2.3.2 Index Optimization

Database indexing is the fastest and easiest way to improve database read operations. As the data scales, depending on the dimensions and type of data growth, indexes should be created and applied to the tables by analyzing slow query logs. For reports, queries may have to be modified to use preferred indexes based on the type of data.

2.3.3 MariaDB / MySQL Scaling

MariaDB / MySQL has multiple strategies to scale including:

1. **Master - Replica:** "Replication enables data from one MySQL server (the master) to be replicated to one or more MySQL servers (the slaves). MySQL Replication is very easy to set up, and is used to scale out read workloads, provide high availability and geographic redundancy, and offload backups and analytic jobs."

2. **MaxScale:** "MariaDB MaxScale is a database proxy that extends the high availability, scalability, and security of MariaDB Server while at the same time simplifying application development by decoupling it from underlying database infrastructure. MariaDB MaxScale is engineered with an extensible architecture to support plugins, extending its functionality beyond transparent load balancing to become, for example, a database firewall."

3. **Galera Cluster:** "Galera Cluster for MySQL is a true Multi-Master Cluster based on synchronous replication. It's an easy-to-use, high-availability solution, which provides high system up-time, no data loss and scalability for future growth."

2.3.4 Data Partitioning

A partition is a division of a logical database or its constituent elements into distinct independent parts. Database partitioning is normally done for manageability, performance or availability reasons, or for load balancing. MariaDB supports partitioning natively and it can be done for large multi year transaction data based on the financial year.

2.4 Scaling the Application Server (Frappe)

Frappe services are Python processes that run in a Python Virtual Environment maintained by **Bench**. The Frappe processes are connected to a database service (MariaDB) and a file system (Ceph Storage).

2.4.1 NGINX Proxy

Since Frappe is designed to be multi-tenant, these services can be run behind a reverse proxy like NGINX and serve multiple sites from a single machine. If we find bottlenecks on the services, we can add machines dynamically to the proxy by creating a new machine running the Bench of the same version connected to the same MariaDB and File System. We can also use Docker containers to quickly deploy Frappe services on the cloud.

2.4.2 Separation of Web and Job Workers

If there are heavy batch processes that need to be executed, the Web and Job Workers can be run on separate machines.

2.4.3 Separation of Read and Write Queries

Frappe Framework provides a configuration that allows separation of read and write queries. If there is a master-replica setup, read requests can be configured to go on the replica servers provided the replica is within acceptable threshold.

2.4.4 Async Python using GEvent

The GEvent library provides an easy way to scale the number of worker processes for communication and database intensive processes. Doing so dramatically increases throughput.

2.5 Scaling Redis

Frappe uses Redis for caching application level summary data that is used frequently. Apart from this, Redis is used to maintain queues for job workers that process scheduled and batched jobs. Redis provides its own clustering system called Redis Cluster.

“Redis Cluster provides a way to run a Redis installation where data is automatically sharded across multiple Redis nodes. Redis Cluster also provides some degree of availability during partitions, that is in practical terms the ability to continue the operations when some nodes fail or are not able to communicate.”

2.6 Scaling Front-end

After crossing a certain threshold, certain UI features become very resource intensive and should be switched off. Frappe Framework provides configuration for switching off resource heavy features like:

1. Open document notifications
2. Count views on UI (on filtering / refresh)
3. Tag Count on list views

3. Application Scaling

While scaling resources can reduce bottlenecks, it is only one dimension of the scaling process. The bigger gains in scaling can be done by going deep into bottlenecks, analyzing them, and applying contextual application level solutions to the problems.

3.1 Performance Monitoring Tools

For all the services, there are many tools available to test and analyze performance. Database bottlenecks can be identified by:

1. Process monitoring to see what processes are taking a long time
2. Logging slow queries that take more than a few seconds to execute
3. Query analysis with the EXPLAIN query helps to understand the number of rows scanned and indexes used.

Python and Frappe services can be analyzed by:

1. Built-in Python Profiler
2. Logging in Frappe

3.2 Scaling Reports / MIS

When the dataset increases, the processing required to generate summary reports like Accounts Receivable, etc., increases. Various strategies can be applied to handle these scenarios

1. **Report Scheduling**: Reports can be scheduled and built at specific time intervals.
2. **Views**: New tables containing the report output can be generated and since Frappe is a metadata framework, these can be filtered and analyzed using the Framework Admin UI.

3. Separate servers can be configured just for processing reports.

3.3 Increasing Throughput

ERPNext is designed to handle various kinds of business scenarios and tries to validate every transaction for various scenarios. If there is a high volume transaction, for example billing, that is created in separate systems and then inserted via API, these can be pre-validated and inserted with a custom API that will bypass ERPNext validations, thereby increasing the rate at which transactions can be inserted into ERPNext.

For extremely high volumes, direct CSV imports in the database can also be done to ensure maximum throughput. These operations will have to be created and maintained separately.

3.4 Refactoring

After hardware and database scaling, there were other areas that could be improved. This included optimizing the code for General Ledger and GSTR reports to reduce the load time.

At times, ERPNext features may not be designed for a particular use case. In certain kinds of businesses, specific data structures exist that make the current feature design extremely resource intensive. In such cases, the best solution would be to analyze the source code and performance using profiling and rewrite a part of it to ensure smooth operations.

This is a hard process that requires domain expertise and understanding of the code architecture. Since ERPNext is an open source project, the opportunity to constantly refactor bottleneck processes exists.

3.5 Partitioning Ledgers / Federation

For large multi-company / branch entities, each of the subsidiary entities can be configured to run a separate instance of ERPNext. These entities could be synchronized using an event based framework like Apache Kafka or built-in Event Streaming support in Frappe (Version 12+).

In such a case, the general accounting and stock ledgers would be maintained separately for each subsidiary and only summary level transactions would be posted in the “central” entity. This would ensure that each entity is scaled independently and the central entity is not burdened with a high volume of consumer transactions or micro actions.

3.6 Customization and Plugins

Frappe Framework has a plugin-architecture that allows these application level configurations and extensions to be maintained and deployed across multiple instances. Most of the “events” in ERPNext can be extended via “hooks” that allow you to add custom tasks to the core event.

These plugins called apps, are maintained in their own git repositories for easy deployment and management.

3.7 Custom Front End

The Frappe “Desk” interface is designed to handle a certain volume and concurrency of data. Beyond a point, certain standardized views that are generated via metadata may become unusable. At this point, custom views can be written that use Frappe REST APIs, this view can be made using any front end stack of choice.

3.8 Load Balancing

Transaction processing can be split into multiple parts and posting of ledgers and other operations can be processed in batches during off peak hours. Batch processes like production planning / MRP run and payroll processing can also be shifted off peak hours. This can increase throughput for the time of high operations load and also lead to better utilization of server resources.

3.9 Migrations at Scale

ERPNext is a fast improving application and it is important to keep patching the application to ensure that new features, performance and security updates are constantly applied. For ensuring smooth migrations, the right customization strategies need to be applied:

1. Avoid core customizations, always use custom apps and hooks to extend functionality.
2. Ensuring unit-testing for all customizations to enable faster upgrade testing.
3. If core customizations are needed, create a parallel process, or ensure the extension is merged in the core product.
4. Namespacing of all extensions so that they don’t conflict with future models.
5. Understanding the roadmap of the project and the maintainers to align the direction of scaling with the core product.

4. Testimonial

“ERPNext is a very cost effective stable Solution for large Implementation.

At VedMata Gayatri Trust, the ERPNext implementation involves more than 500 warehouses, around 13 companies & 40 branches, an average of 6-8 transactions per minute speaks about volume. The modules used include Finance, Manufacture, Inventory, Purchase & Sales, and others.

With the Professional team of Frappe empowering us for mapping with existing systems, they made optimizations to improve performance for a large scale implementation. Most importantly, end Users are really happy with the friendly interface & ease of use.

The professional responses on issues during implementation and optimization (especially on performance) done by the Frappe tech team is really commendable. The optimized cost & 100% availability of the system makes senior management happy.

We surely recommend ERPNext with the professional team of Frappe supporting it for any large scale implementation.”

- Prashant Soni, Volunteer at Shree Vedmata Gayatri Trust

Conclusion

ERPNext is based on off-the-shelf open source components and standardized service oriented web architecture. This makes ERPNext comparatively easy to scale versus legacy systems. A highly scalable deployment would need expertise in each of the individual components and ability to understand and apply the right scaling approach on each bottleneck. These approaches have been laid out in this document and will be required at different points making scaling a multi-disciplinary and high skill job. It is important to either build in-house expertise or to buy services from an experienced service provider for doing it right.

Credits

This document is based on experience by the team at Frappe Technologies which supports thousands of instances of ERPNext. Other than its own internal deployments, Frappe works with several on-premise deployments in the community. There are frequent interactions and exchange of ideas in the community on scaling. For this article, we would like to thank our interactions with two teams, both of whom have large scale deployment and several engineers working on ERPNext. These are Elastic Run, one of the largest full-stack logistics start-ups in India and Zerodha, which is now India's largest retail stock broker. We would also like to thank Vedmata and Selco for presenting us with opportunities to employ scaling techniques. Other than these, the fantastic DevOps team at Frappe Technologies has been working constantly pushing the scale barrier for ERPNext.

Author: Rushabh Mehta

Contact

For further information please connect to the Frappe Technologies team at enterprise@erpnext.com or +91-22-48970555.
Frappe Technologies Private Limited,

D/324, Neelkanth Business Park,

Vidyavihar (W)

Mumbai 400086

India

ERPNext by Frappe

Accounting
Selling
Buying
Stock
Assets
HRMS
Manufacturing
CRM
Assets
Agriculture
Projects
Retail
Non-Profit
QMS
Healthcare
Customization
And More

www.erpnext.com
enterprise@erpnext.com
91-22-48970555

ERPNext is a 100% free and open source software.