**Exploring Sustainable Development and Database Sharding Strategies**

**Prepared For**

Professor: Mufleh-Al Shatnawi

SED600(Topics In Software Design)

Seneca Polytechnic

**By**

Kannav Sethi

Seneca Polytechnic

**28th Jan 2024**

**Introduction**

This report aims to explore the intersection of database sharding and sustainable development practices in addressing the complex scalability challenges faced by modern software systems. **Database Sharding** is a technique that is used to horizontally partition a database into smaller more manageable units called as shards , this addresses the problem of Scalability, Whereas **Sustainable Software Development Practices** tackle the problem of environmental impact of the software systems by introducing sustainable principles into the field of Computer Architecture

**Sustainable Development In Software Engineering**

**Definition and Importance Of Sustainability**

Sustainability is demanded by present-day society to cut down the energy consumption and reduce carbon footprint. Software development technology is no exception and striving to establish smart cities, smart buildings, and a variety of software applications with which to improve sustainability [5].Green and sustainable software engineering is the art of designing and developing green and sustainable software through green software engineering practices [1].The term “green software” refers to the software that emits the fewest carbon emissions [2]. A core set of competencies, i.e. **Principles of Green Software Engineering**, is required to develop, operate, and construct green software[2][3].­There’s evidence to the fact that software on its own doesn’t emit any harmful discharge but the way the software is developed and the way it is used leads to discharge of carbon emissions which can contribute to much larger carbon-footprint associated with it.[4]

Three major activities that are responsible for the reduction of carbon-emissions are

1. **Energy Efficiency**
2. **Hardware Efficiency**
3. **Carbon Awareness**

A diagram of a software principles

Description automatically generated

**Provide reference here -kannav**

This report shall give more information about how to adopt the relevant principles, practices and philosophies to reduce their impact on environment

**Principles**

Green Software Engineering Principles describes the theory that underpins green software, they are the skills essential to develop, create and operate sustainable software applications[2][3].All of these principles are independent of the following factors

1. **Industry**
2. **Programming Language**
3. **Hosting Strategy**
4. **Expertise**

The following points shall cover the principles

1. **Carbon Efficiency:** The goal of this principle is to make applications that have the same value to the users but minimize the carbon emitted per unit work for that application[6][2].
2. **Energy Efficiency:** Energy/Electricity is a proxy for carbon, This goal emphasis on the fact of building energy efficient application to reduce the amount of carbon being emitted by them. **Energy Proportionality** quantifies the relationship between power consumed by a computer and the rate at which useful work is done, this can be one of the factors that can help individuals develop a better understanding of how an application behaves with respect to energy consumption.[2][6][3]
3. **Carbon Awareness:** The energy supplied to a consumer comes from clean (Wind,Hydro,Solar) and dirty (Coal, Fossil Fuels) sources of energy. Clean energy sources have 0 carbon emissions whereas the dirtier sources of energy emit an unhealthy amount of carbon The idea of this principle is to do more work when the energy is coming from clean sources of energy and do less work when the energy comes from dirtier sources. The unit to keep track of the impact of all greenhouse gases is CO2eq (Carbon-Dioxide Equivalent).One of the most important aspect to keep in mind when being carbon aware is Carbon Intensity, this refers to how much carbon is emitted per kilowatt hour of energy consumed (**CO2eq/kWh)**. Carbon intensity is dependent on factors like geographical location and weather conditions , as these impact the availability of cleaner energy sources. The primary objective of this principle is to being aware of the fact that energy consumed doesn’t always have the same impact in terms of carbon intensity.[2][6]
4. **Hardware Efficiency:** Not all the emissions come from the operations of software on a device, carbon is also emitted during the creation and disposal of a device. When the total emission of a device is calculated it is both the running of the device and as well as the embodied carbon that is taken into account for. This principle emphasizes on the fact that to reduce the carbon-emissions from these devices we must be hardware efficient which means we need to **increase the utilisation** of the device and focus on **extending its lifespan**
5. **Measurement :**This principle elaborates on the fact that what you can’t measure , you can’t improve it. The GHG **(Green House Gas)** protocol is one of the most common methods used to measure the total emissions by a software. It takes into account the direct emission by the operation of a company, emission generation of purchased energy and the emission from an organizations supply chain. The GHG aids in calculating the total carbon emissions from an organization but sometimes the GHG fails to explain the full story , that is where the SCI **(Software Carbon Intensity)** metric comes into play. This metric calculates the rate of emission of carbon by a company, it buckets the emissions into **operational emissions** , carbon emissions from running the software and  **embodied emissions,** carbon emissions from the physical sources required to run the software , the only way to reduce SCI score is to be energy efficient, hardware efficient and carbon aware.
6. **Climate Commitments :**This principle emphasises on the importance of an organization committing to various climate-commitments strategies an organization can opt for. This can be being carbon neutral, meaning that for each gram of carbon emitted you do something that removes one gram of carbon from the atmosphere or this could be offsets which are direct investments in the activities that reduce emissions from the atmosphere.

All of the principles stated above act as the backbone of what green software development stands for.

**Philosophies**

There are 2 ideas that drive the Sustainable Software Development, it encourages individuals to see sustainability as a core value. These philosophies are as follows

1. **Everyone has a part to play in climate solution-**every person in the domain of software engineering has a part to play , Sustainable Software Engineering is inclusive
2. **Sustainability is enough to justify the work in itself-**cost effectiveness, faster software and robust system are just added advantages of building a sustainable software, the primary focus should be sustainability.

**Practices**

Sustainable software development involves taking into account several practices that reduce the impact of software on the environment, below are some practices that can be incorporated to the current existing software development processes

1. When doing requirements elicitation and cost analysis, one should try to include the Environmental costs in the document to list out the impact on environment due to the whole **SDLC** (Software Development Life Cycle)
2. Efficient resource management, including the proper allocation of memory, selection of algorithms that complete a task in less amount of time (Having less time complexity) should be taken into account during the **SDLC.**
3. Processes like Spatial Shifting and Temporal Shifting emphasize on the idea of moving to a different region and shifting to your processes to a different time so as to utilise energy when it is coming from lower carbon-emitting sources.
4. To reduce the impact caused by embodied carbon, the amortization of carbon over the expected life span of a device. This means the extension of the lifespan of a device which expands on device utilization
5. A good focus should be given on the developing of system resource usage metrics, this can provide the organization with a metric to track their environmental goals.
6. The usage of cloud based solutions in the development of architecture should be considered as they can commissioned and decommissioned in a matter of hours and would lead to less power utilisation and reduce the “hardware garbage”.
7. The amount of data transferred over the network can lead to a lot of energy consumption, to reduce this what can be done is to implement mechanisms like HTTP Caching which is to store a response associated with a request and reuse the response for subsequent request and Content Delivery Networks (CDN’s), is a geographically distributed group of servers that caches content close to end users.

**Database Sharding In Software Engineering**

**Definition**

Databases stores information in multiple datasets consisting of multiple rows and columns. When the data to be stored in the database become large enough for a single machine to handle , there is a need to optimize it to nullify the negative effect of large data. To tackle this problem **Database Sharding** is implemented. This method is one of the optimization techniques that can be used to solve the problem of large datasets in a single database. This works by splitting a dataset across multiple databases, called **Shards**, which can then be stored on different machines. This method is also termed as Horizontal Scaling , as in contrast to increasing the RAM or CPU power for the machine **(Vertical Scaling)** , it splits the load onto multiple nodes

A blue cylinder with black arrows

Description automatically generated

**Provide reference for this**

**Benefits of Sharding**

Database Sharding has the following benefits:

1. **Scalability/Storage Capacity:** Since the database is being split up into different shards, you can allocate each shard to a different system allowing the shard to use the system’s resources, this allows for more computing resources for each shard thus leading to easy scalability of the system and increase in the capacity of data stored in the database

A diagram of a server

Description automatically generated

**Provide reference here**

1. **Faster Query Response Time:** Compared to the entire database, a shard has less number of datasets in it, therefore due to the nature of a database shard it is quite faster to retrieve a record from shard as you’ll only be searching in a limited small number of datasets. The shards also allow for parallel processing of queries as different shards can simultaneously respond to the requests.
2. **Geographical Distribution:** Sharding allows for easy creation of distributed databases for apps which are geographically distributed , this would allow for the redirection of the request made by the user to the shard that is located near to them. This leads to a reduction of network latency between the requests of the user and the retrieval of data from the shards.

A diagram of a network

Description automatically generated

All of these benefits of **Database Sharding** affect the design of a system as with sharding you also got to take into considerations factors like failure handling, complexities ranging from a single transaction accessing data in multiple shards at once, high availability and efficient resource utilisation.

**Implementation Strategies**

There are 4 different types of sharding rules that can be applied to the shard keys to determine the correct shard node for the row to be allocated to. These rules are as follows :

1. **Range-Based Sharding:** This type of sharding takes a field of the data record as one of the only parameter which based on a predefined range gets allocated to the appropriate node. It would also require the existence of a lookup table for writing or reading all the queries. However, this helps in separation of data records in a table, it might also lead to the overloading of data in a single shard based on the bucket-range provided in the shard.
2. **Hash-Based Sharding:** In this type of sharding you assign the shard key to each row of the table by using a mathematical function called as hash function. The hash value thus produced determines what shard the record goes to. It does provide even distribution of data but if more shards are to be generated then the reassignment of data rows to different shards can be quite complex.
3. **Directory-Based Sharding:** In this type of sharding a lookup-service is used to determine where is a particular piece of data is located , here each data record is assigned a unique identifier which maps to a shard. This is different from range-based sharding as here each data record is stored in a shard based on the developers logic, so its flexible in that way, but if something goes wrong in the lookup-table, the entire structure of shard falls apart, in layman terms, its heavily reliant on the lookup-table
4. **Geo Sharding :** A field containing the user’s geographic location is specified in the main table based on which the sharding technique partitions the data records into different shards , to put it in layman term’s, sharding based on geographic location of the user. This is beneficial in the case when the retrieval of data is concerned with the geographical location of the user , thus decreasing the retrieval time but one of the most important negative factors is that it can put much load on certain regions based on the population whose data is being stored in shard.

**Challenges and Consideration**

Database Sharding assists in providing a lot of advantages to the end-users but with these advantages there are lot of challenges that follows it.

1. **Transactional and Application Complexity :** Most of the times there is not automatic sharding techniques and the developers have to manually shard the database which if not implemented rightly can lead to corruption in data. The management of data coming in from different shard workflows becomes a necessity as not handling them the correct way can lead to wrong information being sent to the user. A transaction request from the server can request data that is being stored on different shards, which would then lead to joining of tables which would lead to high response time from the server and degrade the performance.
2. **Unbalanced Distribution Of Data :** Wrongful implementation of sharding techniques can lead to an uneven distribution of data within the shards, which might lead to one particular shard using much more computing resources than the others leading to a low performance of low throughput of data sent back.
3. **Administration Complexity :**  A single large database only requires maintenance for a single server, in case of sharding, all of the shards have to be managed plus the service that redirects the request to the appropriate shard has to be managed as well.
4. **Infrastructure Costs :** When different shards are being implemented in the organization, much more hardware is also needed, which means much more services are needed to keep them secure. All of these things leads to much larger marginal cost which can sometimes over exceed the cost limits set by the organization.

**Managing Sustainability With The help of Sharding**

Sustainable Development and Database Sharding are very closely related when it comes to providing resource and carbon efficient solution. The practice of sharding leads to solutions that are easily scalable and provide good performance , thus leading to a good energy proportionality which is one of the most important factors when sustainable development is taken into consideration. It also leads to distribution of requests over a network of shards interconnected, which leads to less electricity being consumer for a better service to the end-users. Although, Sharding does assist in achieving the sustainable goals, some disadvantages of sharding might also hinder the progress. Sharding requires much more infrastructure than a normal DBMS would, this would lead to much more carbon emissions in the form of Embodied Carbon , apart from this , we also have the complex operations that are needed to be performed for this which would again require lot of energy in terms of manpower and electricity, which would again conflict with the goal of developing carbon and energy efficient applications.

**\Conclusion**

While database sharding offers significant advantages, it is essential to carefully consider and address its potential environmental and operational impacts to ensure its alignment with sustainable development principles.

**References**

[1] Calero C, Piattini M. *Introduction to Green in Software Engineering, in Green in Software Engineering*. Springer; 2015: 3-27.

[2] *Green Software Practitioner*. Learn Green Software. (n.d.). <https://learn.greensoftware.foundation/>

[3] Westheide, D. (2023, February 13). *What is sustainable software?*. INNOQ.

<https://www.innoq.com/en/articles/2023/02/what-is-sustainable-software/>

[4] *How green is your software?*. Harvard Business Review. (2020, September 1). <https://hbr.org/2020/09/how-green-is-your-software>

[5] Rashid, N., & Khan, S. U. (2018). Agile practices for global software development vendors in the development of green and sustainable software. *Journal of Software : Evolution and Process*, *30*(10), e1964-n/a. <https://doi.org/10.1002/smr.1964>

[6] Zimmergren. (n.d.). *The Principles of Sustainable Software Engineering - training*. Training | Microsoft Learn. <https://learn.microsoft.com/en-us/training/modules/sustainable-software-engineering-overview/>