A Multi-Flow Visual Analytics Web Platform for Detecting High Wind Energy Potential in Urban Environments

Athanasios Koukofikis [1] *, Volker Coors [2]

[1] APCOA ValueSpaces GmbH, Berlinerstr. 80, 13189 Berlin, Germany - athanasios.koukofikis@valuespaces.com
[2] University of Applied Sciences, Schellingstr. 24, 70174 Stuttgart, Germany - volker.coors@hft-stuttgart.de

KEY WORDS: Smart Cities; Visual Analytics; CFD; Wind Simulation; Wind energy; OGC; 3DPS; Web Visualization; Web services; Web Orchestration; WebGL; Distributed Systems.

ABSTRACT:

Smart cities represent a transformative approach to urban development, leveraging advanced technologies to enhance efficiency and sustainability. Moving into the third decade of the 21st century, smart cities are becoming a vital concept of advancement of the quality of life. Without any doubt, cities today can generate data of high velocity which can be used in plethora of applications. The wind flow inside a city is an area of several studies which span from pedestrian comfort and natural ventilation to wind energy yield. We propose a Visual Analytics platform based on a server-client web architecture capable of identifying areas with high wind energy potential by employing 3D technologies and Open Geospatial Consortium (OGC) standards. The assessment of a whole city or sub-regions will be supported by integrating Computational Fluid Dynamics (CFD) outcomes with historical wind sensor readings. The results, in 3D space, of such analysis could be used by a wide audience, including city planners and citizens, for locating installation points of small-scale horizontal or vertical axis wind turbines in an urban area. The implementation presents various flows for spatio-temporal data processing. A number of evaluation plans assess the system performance. The results show an adequate performance, although there is a lot of room for improvement in future work.

1. INTRODUCTION

A smart city may well benefit from the large amount of data collected within its boundaries. Valuable information can be extracted via analytical processes and utilized for achieving sustainability goals. Visualization of data can assist in the analysis process and give better meaningful insights. Visualization has emerged as a new research discipline during the last two decades (Keim et al., 2010) with Visual Analytics being the field of visualization that can provide instant and interactive knowledge to individuals by empowering a flexible control of workflow and information. Three-dimensional visualization is fundamental to establishing a modern information system in an urban space. Urban 3D visualizations of geo-spatial data consist of above surface entities, either physical or artificial, including as basis a Digital Terrain Model (DTM) or the surface of the reference ellipsoid. A typical example is the Digital Twins concept. A 3D model is relayed with data acquired from the physical environment, and a simulation model studies the behavior and the performance of the system. Numerical simulation methods, such as Computational Fluid Dynamics (CFD), have been a successful tool in urban physics research (Blocken, 2015). The 3D spatial representation of a city is widely defined in the research community and geospatial industry by CityGML. CityGML, an Open Geospatial Consortium standard, is based on the Geography Markup Language (GML) (Gröger and Plümer, 2012). CityGML provides a semantic and geometric representation model for city entities as a semi-structured format encoded in XML. By its definition, CityGML can be extended via an Application Domain Extension (ADE), which, among other advantages, provides relevant information for simulation applications. In Helsinki's Kalasatama area, throughout the Digital Twins project (Suomisto et al., 2019), a high accuracy DSM model was generated for visualization purposes and a semantic city model based on CityGML. The CityGML model was used in a CFD simulation in order to investigate the wind impact on the micro climate, comfort, and safety of the streets and pedestrian areas.

The climate in an urban area can play a tremendous role in the socioeconomic status of a city. In particular, the wind flow can influence the planning and morphology of a city, and, at the same time, wind flow is affected by the urban morphology (Franke and Baklanov, 2007). Moreover, the urban morphology in some cases could facilitate the production of wind clean energy since urban and deep water offshore areas remain unharvested, while onshore and shallow water wind farms continue to be studied and realized (Islam et al., 2013, Hand and Cashman, 2017, Borg et al., 2014, Balduzzi et al., 2012, Tabrizi et al., 2014, Mithraratne, 2009, Lee et al., 2018, Ledo et al., 2011). Urban wind energy can provide a decentralized local source of energy for residential areas and reduce the cost of energy by avoiding the losses/costs of long-distance energy transmission (Rezaeiha et al., 2020).

In actuality, there are numerous methods/models to estimate or assess the wind energy in an urban environment, namely on-site measurements, wind tunnel testing (Al-Quraan et al., 2016), Computational Fluid Dynamics, and analytical. Simulation processes, specifically Computational Fluid Dynamics (CFD), play a fundamental role in numerically estimating wind velocity and direction in a segregated/sampled 3D space. Although, in an urban context, a CFD simulation could have a high error tolerance, mainly for performance improvement, the calculation of a dynamic velocity field in the course of a year is greatly demanding. Profiling the wind potential of a 3D city model by utilizing the aforementioned methods cannot be related to additional 3D city models via a geometric similarity factor, i.e., the Hausdorff distance (Hausdorff, 1965), since similarities in the terrain surface model and the wind profile (wind directions, wind speeds) are also required. Monitoring of the wind proper-

This contribution has been peer-reviewed. The double-blind peer-review was conducted on the basis of the full paper.
https://doi.org/10.5194/isprs-annals-X-4-W4-2024-89-2024 | © Author(s) 2024. CC BY 4.0 License.

	Wind Atlases	Statistical Analysis	On-Site Measurement	Wind Tunnel	CFD
Spatial Resolution	low	low/high	high	high	high
Spatial range	large	small/large	low	low	low
Temporal resolution	low	high	high	high	high
Temporal range	high	high	high	low	low
Visual representation	2D	2D/graph	graph	graph	2D/3D/graph
Expertise	low	high	modest	high	high
Accessibility	easy	easy	easy	hard	hard

Table 1. An overview of characteristics of the different wind energy assessment methods.

ties in an urban environment is possible with dedicated anemometers (Beller, 2011) or with weather station bundles which are measuring additional weather variables. Although a large number of such sensors can be installed across the boundaries of a city, interpolation models cannot estimate the wind flow field between the surface model of the city because wind motion obeys to fluid mechanics and is described by the Navier–Stokes equations (Moeng and Sullivan, 2015). A variety of toolsets make use of Computational Fluid Dynamics (CFD) to numerically solve the flow of the wind as a dynamic/time dependent series of vector field snapshots or as a static vector field. Despite the fact that a CFD simulation can model the wind flow in an urban area with better accuracy (Blocken, 2018), these solutions come with high demands/requirements and several restrictions (Table 1).

2. PROBLEM STATEMENT AND MOTIVATION

The evolution in data-driven web-based visualization allows for interactive exploration and information extraction for a wide audience. An interactive visualization, based on 3D city modeling, can extract meaningful information in the field of wind energy. Although, in some cases, a 2D visualization of high wind potential can be facilitated, a 3D representation is more intuitive and provides the opportunity to explore special use cases, such as the integration of wind turbines into the underside of high altitude bridges (Handsaker et al., 2021) or the implementation of an array of vertically aligned wind turbines (Figure 1).

Figure 1. An array of three wind turbines is integrated between the two towers of the Bahrain World Trade Center (by Denise Krebs is licensed with CC BY 2.0.

Identifying locations in an urban environment where a small scale wind turbine could yield an adequate amount of energy in the course of a year is a four-dimensional problem. Three dimensions are used for the spatial component of the location, i.e., the X, Y, Z coordinates in an arbitrary coordinate system where wind speed reaches above a required threshold. The fourth dimension is the time component, i.e., the duration of the required wind speed during a year. A web-based client-server architecture can be used to assess the wind energy potential in an urban environment by integrating static CFD simulations, wind historical data, Web 3D technologies, and OGC Standards, which would allow researchers, entrepreneurs, and civilians to estimate the yearly yield of such investment. A spatio-temporal query in the form of "find locations where the wind velocity is higher than 6 m/s for more than 5 h a day for more than 200 days in a year" is expected to be resolved by such a solution. The results of such query will indicate locations where a small scale vertical axis wind turbine can be installed. The current paper is based on (Koukofikis and Coors, 2021).

Motivated by providing the possibility to the public to assess the wind potential in residential areas, we propose a web-based platform and a dedicated workflow that will give a better insight in interactively exploring simulation data combined with historical readings. Our study investigates if it is possible to design a workflow to analyze and process in near real-time massive simulation data in conjunction with historical readings on the server side combined with a thin visual analytics toolset for eloquent visualizations and data exploration in the client side to give better insights and assist in decision-making in an urban environment. To the best of our knowledge, such a framework is currently not available in the scientific literature. Most of the related research on wind potential focuses on offshore locations, while cases in an urban environment are not interactive, demand high expertise, and Web-based accessibility is absent or limited by 2D visualizations. The varied contributions of our work are: (i) a thin web client with a simplified interface and passive rendering to assist the visual analysis of simulation/historical data in low end desktop and mobile clients, (ii) an interactive integration and processing of simulation/historical data to support an energy sustainable urban environment, (iii) a centralized simulation data architecture with efficient distribution of relevant data controlled via filtering, (iv) a combined data access interface realized by a query API in parallel with the OGC standard 3D Portrayal Service, and (v) a variety of data processing and data transmission schemes.

3. SYSTEM ARCHITECTURE

3.1 Conceptual Design

The proposed approach emphasizes on the investigation of knowledge extraction based on CFD results and historical wind data. The concept is supported by the definition of a web-workflow to assist in locating possible areas in a city environment that small scale wind turbines could be installed. The

workflow is expected to resolve queries formulated as a composite spatial and temporal high-pass filter. Additional functionality should allow the incorporation of wind simulation vector fields for supplementary analysis. To obtain an eloquent visualization, the results of the analysis will be visualized in an interactive Web 3D environment. To facilitate this concept, an integration of historical wind data with CFD simulations is needed. The historical readings, namely timestamp, wind velocity, and wind direction, of a weather station will be used to perform a statistical analysis in order to find the prominent wind directions, i.e., directions with the highest probability. For each prominent wind direction, a second statistical analysis will reveal its prominent wind speeds. Each combination of wind direction and wind speed will become the inflow boundary conditions for a static CFD simulation (Figure 2). The output of each CFD simulation, i.e., a static 3D vector field will get integrated with the historical wind data. The latter will serve as a lookup table. The integration will be realized in several steps: The historical data are queried for similar simulation boundary conditions within a similarity range; then, for each historical entry that qualifies, a matching simulation result will get associated.

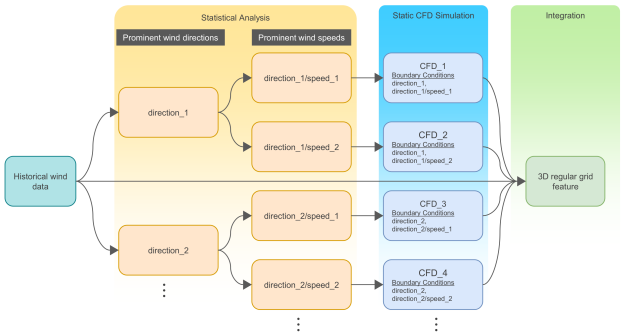

Figure 2. Conceptual graph of the workflow.

The definition of high wind potential locations utilizes a 3D feature layer in the form of a regular three-dimensional grid (Figure 3). The potential is defined in terms of spatial and temporal wind events. Each grid cell captures the spatial information coming from the vector fields of the CFD simulations and the temporal information from the historical wind data. In each grid cell, a reducing function (Equation 1) aggregates the duration and velocity of the wind. The application of a high-pass filter in both wind velocity and duration generates the desired locations in 3D space.

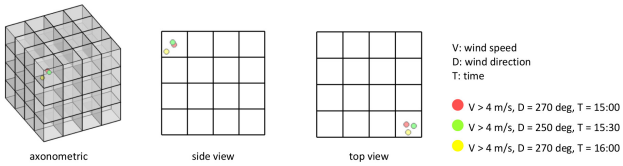

Figure 3. A 3D feature grid with three historical events in a cell. The dots represent the locations of the velocity vectors. The red and yellow vector are results of the same CFD simulation since they are produced from the same wind direction (assuming the same wind speed in the historical data).

$$V_c = \sum_{i=0}^{N-1} \frac{\sum_{j=0}^{S-1} v_j}{S} \left(\frac{t_i}{T}\right), \qquad (1)$$

where V_c = scalar horizontal velocity in a cell
N = number of simulations in a cell
S = number of spatial events in a cell
v = horizontal velocity
t = number of a single simulation temporal events in a cell
T = number of all temporal events in a cell

4. END-TO-END IMPLEMENTATION

An important consideration in our implementation is the need to offer to the end users multiple options to process data. The implemented flows take into consideration the regions of interest, the processing architecture, the data transfer scheme and the web protocol (Table 2).

Flow	ROI	processing	transfer	protocol
flow#1	Focus	centralized	bounded	http
flow#2	Focus	centralized	stream	ws
flow#3	Area	centralized	bounded	http
flow#4	Area	distributed	stream	ws

Table 2. The end-to-end implemented flows for spatio-temporal data processing.

4.1 Frontend Implementation

In order for the client side to fulfill its scope, the following components are needed:

- A 3D city model to support the visualization since it can give a better perspective in identifying city areas and relative locations between buildings and high potential areas. It can also assist in reasoning in cases of low potential areas because of building occlusion.
- An interactive formulation of queries for investigating a city area in an easy manner, allowing multiple, adjustable and repeating requests.
- Definition of time and space properties is an essential part of a query, as mentioned before, since we want to apply filtering in four dimensions, i.e., three dimensions for space and one dimension for time.
- Augmentation of the 3D city models with the query results gives a better insight of the potential of an area in three space dimensions. The capabilities of the 3D engine enables an interactive navigation in all areas that might show high potential.
- Two types of regions of interest (ROI), i.e., “Focus ROI” and “Area ROI” each one with spatial constraints.
- Different processing flows for each region of interest, i.e., centralized and distributed processing.
- Different data transaction approaches, i.e., bounded and streaming.
- A multi-layer visualization design can further assist or reason the analysis results. Information coming from separate layers, where query results are stored, can be combined in the same 3D scene to better empower the analysis.

On the client side, the end-users are able to navigate and visually analyze either the integration of the historical wind data with the CFD simulation data, as well as simulation results, utilizing a multi-layer scheme. The 3D module of the web client is based on Three.js, a powerful JavaScript 3D engine bringing computer vision to the browser. The web client is based on the Flux Architecture, which is a unidirectional flow of control and information. For that purpose, the redux library is controlling the state management of the web application. The backbone of the client is realized in Riots.js, a library which shares commonalities with React.js. It is a popular client-side framework/library, offering custom HTML tags, which are the building blocks of the user interface. Each custom tag could be described as a tiny MVC application. The user interface is composed by a Single Page Application. The layout is separated into two main sections: the 3D Engine viewer covering the whole screen and the side-bar on the left side of the screen. The 3D viewer is the rendering component of 3D static assets, i.e., buildings, terrain, and dynamically generated query results or CFD attributes. Additionally, the user can interact with the 3D viewer, via defining clipping planes, in order improve the visualization of results in specific city volumes. The side-bar is a scroll-able generic area of graphical user interface to control the workflow. Various user interface controls are logically grouped and wrapped in collapsible graphical elements in order to save space on the side-bar.

The first group of user interface controls in the side-bar defines the Regions of Interest (ROI). A Region of Interest defines the 3D spatial domain as a tight bounding volume in order to exclude CFD data (e.g., wind velocity 100 m above ground when the highest building is 30 m) and improve performance. The creation of an ROI is interactive. A defined ROI can be reused multiple times in the analysis workflow. There are two ROI definition modes: (a) Focus mode and (b) Area mode. An ROI in focus mode is used to analyze small areas in a city. It has maximum size constraints of 50 m in all dimensions (width, length, height). A typical example of usage could be the potential analysis of a single building. An ROI in area mode (Figure 4) is used to analyze larger areas in a city. It has minimum size constraints of 100 m in planar dimensions (width, length).

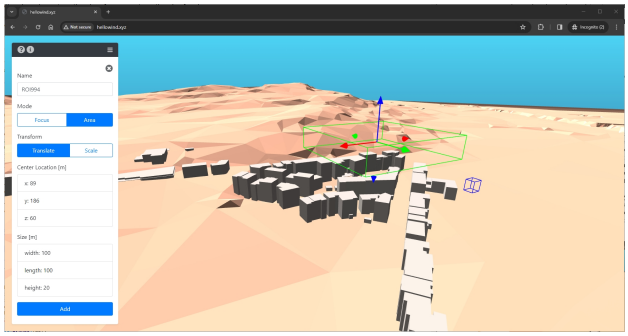

Figure 4. Definition of an ROI in area mode (green box). In blue an already defined ROI in focus mode.

A focus ROI is related to two processing flows, i.e., flow#1 and flow#2 (Table 2). Data are processed in a single computing node but the transfer scheme can be bounded or streaming. On the other hand, an area ROI can access a centralized processing flow or a distributed processing flow.

4.2 Backend Implementation

At the heart of the data tier is a PostgreSQL database, a powerful database management system. The relational database public schema of our implementation (Figure 5) defines the tables for storing the historical wind data, the boundary conditions for the performed CFD simulations, the resulting 3D vector fields and the descriptive statistics for each variable of a single simulation. The latter will become useful information when applying value filtering in the client. Additionally, there are several temporary tables, which do not belong to the public schema, which are part of the interactive processing of the resulting 3D grid. PostgreSQL manages transactional integrity by using a Multiversion Concurrency Control model (MVCC). Therefore, every transaction in the database is realized in its own context.

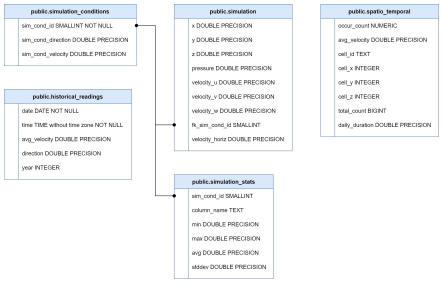

Figure 5. The public schema of the PostgreSQL database.

The spatio-temporal grid generation takes place in the database system. It is segmented in several database procedures and functions. Several functions and procedures perform spatial and temporal filtering. The relations between those routines is depicted in Figure 6.

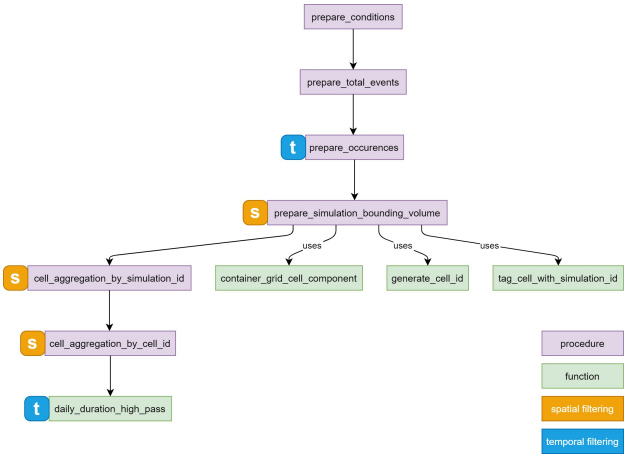

Figure 6. The series of steps of the spatio-temporal grid generation.

The distributed flow (Table 2) is based on BullMQ (Taskforce.sh, 2024), a message queue system built on top of Redis. A message queue is a messaging middleware that enables applications and services to communicate in a decoupled manner. Operating as intermediary buffers, message queues store and manage messages until they are consumed by the intended recipients. Redis (Redis, 2024) is a well known in memory key-value data store with significant performance, high throughput and low latency. It's data model supports strings, hashes, lists, sets, and more, which makes it a versatile tool for data processing in a distributed system. The communication

This contribution has been peer-reviewed. The double-blind peer-review was conducted on the basis of the full paper.
https://doi.org/10.5194/isprs-annals-X-4-W4-2024-89-2024 | © Author(s) 2024. CC BY 4.0 License.

protocol between client and server is WebSockets. The client requests an action under a specific namespace which acts as a communication channel with a spatio-temporal query payload. The server breaks down the bounds parameter of the query into batches (Figure 7) using a 3D bin packing algorithm (Wu et al., 2010).

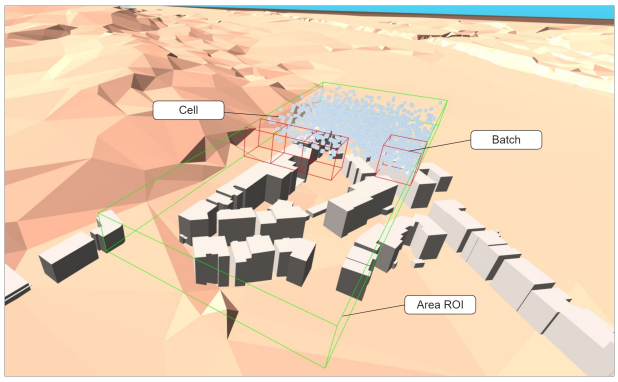

Figure 7. A region of interest is split into tight fit batches. The result in each batch is processed by a BullMQ worker.

Using the BullMQ models (queue, job), two queues are created. The first queue will manage the parent jobs and the second queue the children jobs. All batches are converted to children jobs which in turn become the children of a single parent job. Once the parent job is created, the BullMQ workers keep requesting tasks from the message queue server until the queue gets empty. When a child job is resolved, the results are transmitted back to the client through a WebSockets connection where the data get rendered. When all children jobs are resolved, the parent job is resolved and the socket connection can close.

To improve fault tolerance, scalability, and data availability in our distributed system, database replication is used. Although at runtime the database system is read-only, replication increases the availability of historical and CFD data to all workers (Figure 8).

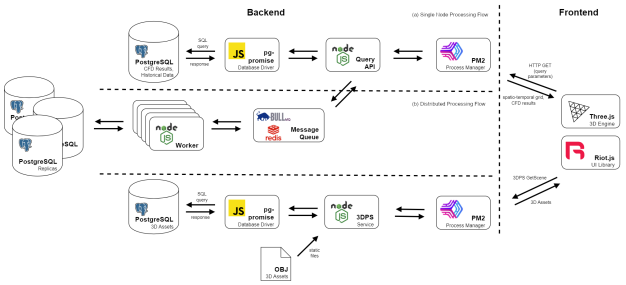

Figure 8. The end-to-end implementation components divided into frontend and backend tiers.

The access of the 3D assets, namely terrain and buildings, is abstracted via OGC's 3D Portrayal Service. The 3D Portrayal Service (Hagedorn et al., 2017) is an OGC Standard that abstracts the access of 3D geospatial datasets in various client platforms via the web mainly for visualization purposes. The standard specifies three methods to access information: GetCapabilities, AbstractGetPortrayal, and AbstractGetFeatureInfo. A GetCapabilities request returns information about the available request methods, the extents of the data, data layers (buildings, vegetation, etc.), layer styles, and streaming formats supported. The retrieval of a scene is implemented by the AbstractGetPortrayal operator. The GetScene method, which is realized in our solution, is used for client-side rendering and the GetView method for server-side rendering. The 3D Portrayal Service specification does not indicate a specific content delivery format. Among several payload formats, OGC community standards which expose a bounding volume hierarchy, namely 3D-Tiles (Cozzi et al., 2018) and I3S (ESRI, 2019), can be served by a 3DPS GetScene implementation.

4.3 Streaming Database

Although streaming databases offer a variety of solutions, legacy database systems can mimic streaming capabilities by employing generator functions. Generators in JavaScript (MDN, 2024) are stepwise versatile functions which can be paused and resumed. This provides a plethora of possibilities for synchronous and asynchronous workflows. Establishment of communication between the frontend client and the backend is achieved via WebSockets (Figure 9). In particular, the library Socket.IO (Socket.IO, 2024) is used (both client and server implementations). The processing flow#2 (Table 2) utilizes this technique with PostgreSQL v12.17, a database management system version which does not support streaming. A spatio-temporal query is passed in the constructor of the generator which defines the bounds of the focus Region of Interest. The generator delegates the spatio-temporal query for each cell inside the Region of Interest, yields the result and stops execution waiting for the next Socket.IO event to process the next cell.

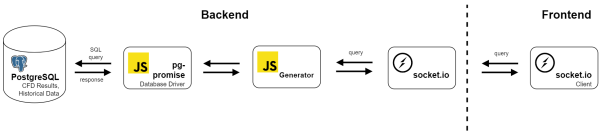

Figure 9. Utilizing a generator to stream data from a database.

5. EVALUATION

Our visual analytics platform supports query processing via various scales of region of interest (focus, area) and flows (centralized, distributed). To evaluate the interactiveness and performance of the flows of our platform and investigate the parameters which greatly affect the response times, we facilitate a number of evaluation plans. Each plan is designed to perform a spatio-temporal query to area regions of interest for both centralized and distributed flows. All the query parameters are constant except the bounds which is the parameter that is varying (Table 3, plan#1 and plan#2). In addition, the performance of the distributed system is examined with variable number of workers (Table 3, plan#3). We utilize OpenStack (OpenStack, 2024) as test environment to provide computing nodes with different specifications (Table 4). OpenStack is an open-source cloud computing platform that facilitates the creation and management of both public and private clouds. OpenStack Nova is the component serving as the primary compute engine responsible for managing and provisioning virtual machines (VMs) and instances.

5.1 Centralized and distributed processing

This evaluation plan makes a comparison between a main node which employs centralized processing and a distributed processing system of 8 workers with specs shown in Table 4. Each evaluation is defined as:

This contribution has been peer-reviewed. The double-blind peer-review was conducted on the basis of the full paper.
https://doi.org/10.5194/isprs-annals-X-4-W4-2024-89-2024 | © Author(s) 2024. CC BY 4.0 License.

Plan	bounds	batch size	cell size	no. of workers
#1	varying	20 m	2 m	8
#2	varying	10 m	2 m	8
#3	varying	20 m	2 m	[2, 4, 6, 8]

Table 3. The evaluation plans and the parameter values. Batch size and number of workers are only applicable to the distributed flow.

Specification	Main Node	Worker
OS	Ubuntu 20.04	Ubuntu 20.04
VCPU	AMD EPYC	AMD EPYC
VCPU Base Freq.	2.4 Ghz	2.4 Ghz
No. of VCPUs	4	1
RAM	16GB	2GB
Database	PSQL v12.17	PSQL v12.17

Table 4. Software and hardware specifications of the testing environment.

$$(b_i, bs, cs, w) \rightarrow (t_i, f_i, o_i) \quad \text{for } i \in [n], \tag{2}$$

where
b_i = bounds
bs = batch size
cs = cell size
w = number of workers
t_i = response time
f_i = initial frame time
o_i = network overhead time
n = number of varying bounds

The bounds for each evaluation plan are defined as offsets in reference to the coordinate system origin:

$$b_i = (lt_i, rt_i, bk_i, ft_i, bm_i, tp_i) \quad \text{for } i \in [n], \tag{3}$$

where
lt_i = left offset
rt_i = right offset
bk_i = back offset
ft_i = front offset
bm_i = bottom offset
tp_i = top offset
n = number of varying bounds

The initial frame is used as an indicator of the responsiveness of the distributed system when a request is sent. In simple terms it indicates when the processing result of the first batch is returned. All evaluation plans (1, 2 and 3) use regions of interest which cover areas from 4 to 484 square decameters (dam^2).

The first evaluation (Figure 10) compares the response time between the centralized and distributed processing when the batch size is 20 meters. A second vertical scale is used to capture the network overhead in milliseconds. Results show that centralized processing consistently outperforms the distributed. Centralized processing response time stays below 2 seconds through all areas. Although the response time in the distributed system increases linearly, the initial frame is comparable to the centralized system when the bounded area is small and remains almost constant regardless the of the size of the bounded area. The network overheads in both systems evolve differently but at the same time do not present a performance bottleneck.

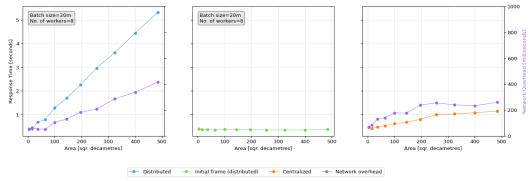

Figure 10. Response time comparison between centralized and distribute processing. Batch size in distributed processing is 20 meters.

The second evaluation plan (Figure 11) depicts the performance impact of the distributed system when the batch size decreases to 10 meters. It is obvious that the decrease of the batch size into half drastically downgrades the performance and increases the network overhead while at the same time keeping the initial frame similar to 20 meters batch size.

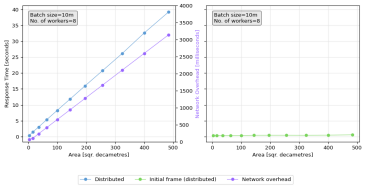

Figure 11. Response time when the batch size in distributed processing is 10 meters.

The third evaluation (Figure 12) compares the response time of the distributed system when the number of workers variate. The results show an almost linear increase in performance when the number of workers doubles.

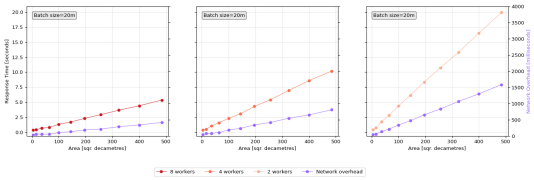

Figure 12. Response time comparison in distributed flow between varying number of workers.

5.2 Cognitive complexity between WebSockets and Streams

Cognitive Complexity is a language-neutral metric that applies to source code structures (classes, objects, functions, etc.) to measure the mental, or cognitive effort required to understand those structures (Campbell, 2018). In this particular case two listings are compared, each one is a frontend client that tries to read the same set of data coming from the backend (Figure 13). The WebSockets approach is utilizing the flow described in section 4.3. On the other hand, the streams approach is based on

Node.js Readable streams. To calculate the cognitive complexity for each approach the static code analysis tool SonarCube is used.

```
socket.on("stream:result", (cell) => {
  try {
    if (cell.valid) {                    // +1
      renderCell(cell);
    }
  } catch (error) {                      // +1
    console.log(error);
  }
  socket.emit("stream:get");
});
```

```
axios.get(url, { responseType: "stream" }).then((response) => {
  let stream = response.data;
  let scratchChucks = "";

  stream.on("data", (chunk) => {
    let decodedChuck = textDecoder.decode(chunk);

    if (decodedChuck.slice(-1) != "\n") {          // +2 (nesting=1)
      scratchChucks += decodedChuck;
    } else {                                       // +1
      scratchChucks += decodedChuck;
      let chunks = scratchChucks.slice(0, -1);
      scratchChucks = "";
      let cells = chunks.split("\n");
      for (let cell of cells) {                    // +3 (nesting=2)
        try {
          renderCell(JSON.parse(cell));
        } catch (error) {                          // +4 (nesting=3)
          console.log(error);
        }
      }
    }
  });

  stream.on("end", () => {
    console.log("ended");
  });
});
```

Figure 13. Cognitive complexity calculation for the WebSockets listing (left) and the Node.js Readable Stream listing (right). The comments indicate the location and the score of the complexity.

The result, sockets=2; streams=10, shows a much lower cognitive complexity in the sockets example. Noticeable is the fact that a stream's outbound buffer captures no semantics. This is the reason the streams approach makes use of Newline delimited JSON (Hoeger et al., 2014), even though there is no guarantee that on the read event the frontend client will receive a single cell encoded as NDJSON. Although the cognitive complexity of the backend implementation of both approaches is not considered, it is worth mentioning that Readable/Writable streams in Node.js give an option to the implementer to apply back-pressure to a stream when a producer or a consumer speed is lower than the respective consumer/producer of the stream. This added consideration is not applicable to the WebSockets paradigm since the producer (backend) syncs with the request rate of the consumer (frontend).

6. CONCLUSIONS

In this study, an interactive visual analytics platform was presented that is able to locate areas in an urban environment where small scale wind turbines could be installed. The architecture is based on community best practices, and its implementation utilizes standardized web and 3D technologies. The visualization on the client side is based on a number of RESTful APIs. Although the number of backend services is small, the absence of an API gateway and service orchestration could affect the overall system reliability. Presently, a form of orchestration is facilitated in the frontend implementation. It is important to mention that the query parameter dailyDuration, of the query API, is a normalized value in the course of a year and does not guarantee the daily duration of the wind above the horizontal velocity threshold on daily basis.

The performance of various processing flows supported by the platform was evaluated against the spatial domain volume, including various scenarios. In limited spatial domains the preferable flow characteristics, i.e., either centralized or distributed processing or either bounded or streaming data transfer plays insignificant role performance wise. When the region of interest becomes large enough to cover the bounds of a city the preferred flow plays significant role. The centralized processing of the spatio-temporal data outperformed the distributed system. Namely, the reasons can be the limited number of workers and the specifications of a single worker. A great performance drop was detected when the batch size decreased, which suggests that an optimal batch size is required in a distributed system realization. Similar was the performance drop when the number of workers decreased. Thus, it is important to identify the requirements and the edge cases of a web application before any architectural decisions. A distributed system strategy can end up being an over-engineered solution and maintenance bottleneck. A centralized system facilitates easier monitoring, configuration, and maintenance. On the other hand, a distributed system has the advantage of improved fault tolerance and reliability, as it reduces the risk of system failure due to a single point of failure. Additionally, distributed systems can scale more effectively by spawning additional nodes as needed, enhancing performance and accommodating increasing user demands.

Although the ground area of the domain volume reached almost 500 dam^2, the response time of the centralized flow and the initial frame of the distributed flown stayed below 2 seconds. One of the reasons is the utilization of spatial indexing, one among other spatial capabilities of the PostgreSQL database system. PostgreSQL spatial indexing depends on the PostGIS extension, a data-driven R-Tree (Guttman, 1984) structure which uses the idea of a spatial containment relationship. Although spatial indexes aim for the query planning improvement, there are cases where this may result in efficiency degradation, which may adversely affect query performance. Counter-intuitively, it is not always faster to do an index search: if the search is going to return every record in the table, traversing the index tree to get each record will be slower than sequentially reading the whole table (Ramsey, 2021).

Partitioning the data regarding the CFD results will considerably decrease parts of the database queries where joins are performed, since the the first step of any database join is a cartesian product. It is obvious that the centralized flow will reach its performance limits while the query bounds increase. For this reason, a streaming design is favorable for achieving (near) real-time results. Expanding the search domain in higher geographical and administrative regions would require the streaming of 3D assets as hierarchical 3D delivery formats, namely 3D-Tiles and I3S, which are widely adopted OGC community standards.

The client application can be accessed at http://hellowind.xyz. A wind energy yield calculation module is already implemented but not performance wise optimized.

Acknowledgments

This work was performed within the interdisciplinary joint graduate research training group Windy Cities in corporation with the University of Stuttgart and the University of Applied Sciences Stuttgart. The computational resources were provided by the bwCloud, funded by the Ministry of Science, Research and Arts and the universities of Baden-Württemberg, Germany. The authors would like to thank the PhD students Maximilian von der Grün for conducting the wind flow simulations and Shubhi Harbola for performing the statistical analysis and pattern recognition of the historical wind data. Both students are participants in the Joint Graduate Research Training Group Windy Cities.

Nomenclature

3DPS	3D Portrayal Service
API	Application Programming Interface
CFD	Computational Fluid Dynamics
CityGML	City Geography Markup Language
CPU	Central Processing Unit
DSM	Digital Surface Model
DTM	Digital Terrain Model
glTF	Graphics Language Transmission Format
HTTP	Hypertext Transfer Protocol
WS	WebSocket
I3S	Indexed 3D Scene
MVC	Model View Controller
MVCC	Multiversion Concurrency Control
OGC	Open Geospatial Consortium
REST	Representational State Transfer
SQL	Structured Query Language
VM	Virtual Machine

REFERENCES

Al-Quraan, A., Stathopoulos, T., Pillay, P., 2016. Comparison of wind tunnel and on site measurements for urban wind energy estimation of potential yield. *Journal of Wind Engineering and Industrial Aerodynamics*, 158, 1-10. https://www.sciencedirect.com/science/article/pii/S0167610516301246.

Balduzzi, F., Bianchini, A., Carnevale, E. A., Ferrari, L., Magnani, S., 2012. Feasibility analysis of a Darrieus vertical-axis wind turbine installation in the rooftop of a building. *Applied Energy*, 97, 921-929. https://www.sciencedirect.com/science/article/pii/S0306261911008026. Energy Solutions for a Sustainable World - Proceedings of the Third International Conference on Applied Energy, May 16-18, 2011 - Perugia, Italy.

Beller, C., 2011. Urban Wind Energy. PhD thesis.

Blocken, B., 2015. Computational Fluid Dynamics for urban physics: Importance, scales, possibilities, limitations and ten tips and tricks towards accurate and reliable simulations. *Building and Environment*, 91, 219-245. https://www.sciencedirect.com/science/article/pii/S0360132315000724.

Blocken, B., 2018. LES over RANS in building simulation for outdoor and indoor applications: A foregone conclusion? *Building Simulation*, 11, 1-50.

Borg, M., Shires, A., Collu, M., 2014. Offshore floating vertical axis wind turbines, dynamics modelling state of the art. part I: Aerodynamics. *Renewable and Sustainable Energy Reviews*, 39, 1214-1225. https://www.sciencedirect.com/science/article/pii/S1364032114005486.

Campbell, G. A., 2018. Cognitive complexity: an overview and evaluation. *Proceedings of the 2018 International Conference on Technical Debt*, TechDebt '18, Association for Computing Machinery, New York, NY, USA, 57–58.

Cozzi, P., Lilley, S., Getz, G., 2018. 3D Tiles Specification. https://github.com/CesiumGS/3d-tiles/tree/master/specification.

ESRI, 2019. I3S Specification. https://github.com/Esri/i3s-spec.

Franke, J., Baklanov, A., 2007. *Best Practice Guideline for the CFD Simulation of Flows in the Urban Environment: COST Action 732 Quality Assurance and Improvement of Microscale Meteorological Models*.

Gröger, G., Plümer, L., 2012. CityGML – Interoperable semantic 3D city models. *ISPRS Journal of Photogrammetry and Remote Sensing*, 71, 12-33. https://www.sciencedirect.com/science/article/pii/S0924271612000779.

Guttman, A., 1984. R-trees: A dynamic index structure for spatial searching. *Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data*, SIGMOD '84, Association for Computing Machinery, New York, NY, USA, 47–57.

Hagedorn, B., Thum, S., Reitz, T., Coors, V., Gutbell, R., 2017. 3D PORTRAYAL SERVICE version 1.0. http://docs.opengeospatial.org/is/15-001r4/15-001r4.html.

Hand, B., Cashman, A., 2017. Conceptual design of a large-scale floating offshore vertical axis wind turbine. *Energy Procedia*, 142, 83-88. https://www.sciencedirect.com/science/article/pii/S1876610217357417. Proceedings of the 9th International Conference on Applied Energy.

Handsaker, S., Ogbonna, I., Volkov, K., 2021. CFD Prediction of Performance of Wind Turbines Integrated in the Existing Civil Infrastructure. *Sustainability*, 13(15). https://www.mdpi.com/2071-1050/13/15/8514.

Hausdorff, F., 1965. *Grundzuege der Mengenlehre*. Chelsea.

Hoeger, T., Dew, C., Pauls, F., Wilson, J., 2014. Newline delimited JSON. https://github.com/ndjson/ndjson-spec.

Islam, M., Mekhilef, S., Saidur, R., 2013. Progress and recent trends of wind energy technology. *Renewable and Sustainable Energy Reviews*, 21, 456-468. https://www.sciencedirect.com/science/article/pii/S1364032113000312.

Keim, D., Kohlhammer, J., Ellis, G., Mansmann, F. (eds), 2010. *Mastering the information age : solving problems with visual analytics*. Goslar : Eurographics Association.

Koukofikis, A., Coors, V., 2021. A Visual Analytics Web Platform for Detecting High Wind Energy Potential in Urban Environments by Employing OGC Standards. *ISPRS International Journal of Geo-Information*, 10(10). https://www.mdpi.com/2220-9964/10/10/707.

Ledo, L., Kosasih, P., Cooper, P., 2011. Roof mounting site analysis for micro-wind turbines. *Renewable Energy*, 36(5), 1379-1391. https://www.sciencedirect.com/science/article/pii/S096014811000501X.

Lee, K.-Y., Tsao, S.-H., Tzeng, C.-W., Lin, H.-J., 2018. Influence of the vertical wind and wind direction on the power output of a small vertical-axis wind turbine installed on the rooftop of a building. *Applied Energy*, 209, 383-391. https://www.sciencedirect.com/science/article/pii/S0306261917312126.

MDN, 2024. Iterators and generators. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Iterators_and_Generators.

Mithraratne, N., 2009. Roof-top wind turbines for microgeneration in urban houses in New Zealand. *Energy and Buildings*, 41(10), 1013-1018. https://www.sciencedirect.com/science/article/pii/S037877880900098X.

Moeng, C.-H., Sullivan, P., 2015. Numerical models — large-eddy simulation. G. R. North, J. Pyle, F. Zhang (eds), *Encyclopedia of Atmospheric Sciences (Second Edition)*, second edition edn, Academic Press, Oxford, 232–240.

OpenStack, 2024. OpenStack Overview. https://www.openstack.org/software/.

Ramsey, P., 2021. Spatial indexing - introduction to postgis. https://postgis.net/workshops/postgis-intro/indexing.html.

Redis, 2024. Introduction to Redis. https://redis.io/docs/about/.

Rezaeiha, A., Montazeri, H., Blocken, B., 2020. A framework for preliminary large-scale urban wind energy potential assessment: Roof-mounted wind turbines. *Energy Conversion and Management*, 214, 112770. https://www.sciencedirect.com/science/article/pii/S0196890420303083.

Socket.IO, 2024. Socket.IO Introduction. https://socket.io/docs/v4/.

Suomisto, J., Airaksinen, E., Bergstrom, M., Heinonnen, H., Lahti, K., Kaisla, K., 2019. The Kalasatama Digital Twins Project. Technical report, Helsinki 3D+.

Tabrizi, A. B., Whale, J., Lyons, T., Urmee, T., 2014. Performance and safety of rooftop wind turbines: Use of CFD to gain insight into inflow conditions. *Renewable Energy*, 67, 242-251. https://www.sciencedirect.com/science/article/pii/S0960148113006113. Renewable Energy for Sustainable Development and Decarbonisation.

Taskforce.sh, 2024. What is BullMQ. https://docs.bullmq.io/.

Wu, Y., Li, W., Goh, M., de Souza, R., 2010. Three-dimensional bin packing problem with variable bin height. *European Journal of Operational Research*, 202(2), 347-355. https://www.sciencedirect.com/science/article/pii/S0377221709003919.