# Research Statement

Isam Mashhour Hasan Al Jawarneh

My research generally focuses on several aspects of big data management, distributed computing, and software architectures for data-intensive applications, with an emphasis on geo-referenced and temporal (including time-series) data. I also work with optimization problems in distributed computing systems and approximation algorithms for smart cities and mobility informatics.

1. **Introduction and current work**

   The term *big data* has taken so far most of the attention of the community in the related art with 3Vs characterizing the data (i.e., volume, velocity, and variety). This, however, does not include spatial or temporal awareness. Most big data coming from IoT, however, are characterized by being geo-referenced (spatial) and temporal. Most Cloud-based Data Stream Processing (DSP) engines are unaware of spatial or temporal features of data, leaving the handling of those aspects as logistics to be considered in the application layer, which overburdens the shoulders of front-end developers with complexities that distract their focus far away from the main objectives of their software design patterns, thus, unnecessarily delaying the overall production process. Even though few relational and non-relational DBMSs, in addition to Cloud-based NoSQL and MapReduce-based ecosystems are incorporating transparently supports for temporal and spatial data management, the field is still at its infancy. Also, most supports are purpose-built. Stated another way, temporal databases have weak support for spatial analytics and vice versa. However, interesting real-world scenarios necessitate the awareness of both altogether. I argue that today it is necessary to focus on providing and transparently incorporating both aspects in a single framework that supports spatial and temporal analytics together such that they both synergistically reinforce each other without their limitations. Also, designing spatial (and likewise temporal) databases optimizations such as indexing (caching etc.,) algorithms for the distributed computing world are confronted with obstacles that do not normally affect their server-based counterparts in the same way. The latter are designed to run in single servers, making it challenging to design distributed analogous which may require significant works and rethinking.

   Having said that, current Cloud-based big data architectural models are general-purpose and not suitable in their current shapes for spatial, temporal, or times-series analysis. For example, the Lambda architecture [1] for big data management is a general model that employs real-time stream processing (in a layer termed as speeding layer) for timely approximate results and batch processing (in a batch layer) for delayed accurate results. However, Lambda architecture is a general architecture that is not attunable with the nature of data that is arriving from heterogeneous IoT data streams, rendering its adoption as-is inappropriate for spatially-laden scenarios. Lambda architecture is not a panacea however, but otherwise serves as a simple place that has inspired me during my Ph.D. to consider it as a springboard for designing a novel architectural model for spatial data streaming loads. Specifically, during my Ph.D., I have designed SpatialDSMS [2] (short for Spatial Data Stream Management System), a fully functional QoS-aware data management framework for big geo-referenced data coming in fast streams from IoT and sensor-enabled devices in dynamic application scenarios such as those in smart cities.

2. **An Architectural model for spatial Cloud-based big data management with QoS guarantees: *SpatialDSMS***

   Current architectural models for big data management in the Cloud are not focusing on performance analysis under uncertainty. This problem is exacerbated when feeding those systems with spatiotemporal data. Therefore, in their stock versions, they are not able to meet stringent

performance requirements imposed normally by dynamic application scenarios in smart cities and urban computing. This normally leads to antipatterns that devalue the promise and benefits of parallel distributed computing, causing undesirable behaviors. By designing SpatialDSMS as an architectural model for spatial big data management under uncertainty in the Cloud, I basically aimed at enabling spatial-aware model performance analysis as a first-class citizen incorporated transparently within the layers of de facto standard big data management frameworks that span all the stages of big data management in the Cloud (i.e., scalable storage, batch in-memory and stream processing). Working on three synergistically collaborating layers analogous to those recommended in Lambda architecture but incorporating spatial-awareness in every single aspect that may otherwise cause antipatterns if left untreated. SpatialDSMS constitutes SpatialBPE [3-5] (short for Spatial Batch Processing Engine) for spatial batch in-memory processing, SpatialNoSQL [6, 7] for scalable distributed storage of spatial data in the Cloud, SpatialSPE [8, 9] ( stands for Spatial Stream Processing Engine) for spatial data stream processing, and SpatialSSJP [10] (short for Spatial Stream Static Join Processor) for efficiently joining transient highly skewed spatial data (i.e., having a fluctuating arrival rates) with data-at-rest. The contribution of SpatialDSMS is that it avoids known pitfalls (gathered by appropriately surveying the relevant state-of-art, documented in our survey [11]) that may lead to antipatterns when applying big data systems in smart city dynamic application scenarios. In the survey, we have identified the most recurrent Cloud-related performance antipatterns that are specific to spatial big data loads. For example, spatial data locality (SDL) preservation can be loosely defined as the ability of data partitioning methods to co-locate the geometrically-nearby geospatial objects while distributing them to parallelly connected Cloud computing resources. SDL has not been adequately considered by traditional architectural models, leading to antipatterns that degrade system performance to points that may potentially bring system into a halt at times.

Specifically, I have enriched the architecture with QoS-aware optimizations that are attuned to the nature of the arriving data streams (spatial in this case). I have injected the QoS and spatial awareness transparently within my architecture so that the user in the presentation layer benefits from the overall optimizations provided without the need to reason about the logistics in the underlying layers.

In short, SpatialDSMS has been designed with five main architectural design goals: modularity, elasticity, dependability, composability and QoS guarantees. The four main parts of SpatialDSMS are operating synergistically for serving complex mixed workload scenarios coming from dynamic environments. With that in mind, modularity is an important attribute that enables all submodules to be plugged in/out in a hot swappable fashion. All constituent parts of SpatialDSMS are patches and glues that compile down to appropriate abstractions of the codebases of the underlying systems atop which they are glued, thus exploiting underlying functionalities without additional efforts and leaving logistics handling to codebases of the underlying ecosystems. In addition, the baselines that I provided can be combined collectively (achieving a desired composability degree), or in a mashup fashion to solve most interesting highly dynamic application scenarios. Also, SpatialDSMS is dependable in the sense that it efficiently handles scenarios with oscillating and fluctuating data arrival rates (a performance parameter) that normally exhibit temporal skewness, specifically for highly dynamic and scalable application scenarios fully loaded with multidimensional geospatially-tagged workloads. Furthermore, SpatialDSMS offers a variety of operation modes, ranging from exact closed-form solutions to probabilistic approximations, depending on the application scenario aiming at achieving a prespecified list of quality constraints including a tradeoff between the result's accuracy and latency/throughput. Moreover, SpatialDSMS system runs within the boundaries of the specified budget expressed as latency/throughput and accuracy QoS guarantees goals. In addition to maximizing the resource utilization.

To prove the robustness of SpatialDSMS, I have designed a model-based framework that incorporates SpatialDSMS as a modelling stage followed by performance analysis and refactoring stages. I have termed my framework as ***cause/effect-tactic-measure*** (refer to section 3.2.1 of my Ph.D. thesis for more details), which is specially designed for spatially-attuned QoS awareness. It simply operates by first identifying the *causes* and *effects* of antipatterns on spatial big data scenarios, thereafter, QoS-aware *tactics* are applied for mitigating the effects (i.e., reversing them). *Measures* then are imposed to measure the ability of every approach (i.e., from the tactics) in achieving the QoS goal. To the best of my knowledge, SpatialDSMS is unique in its ability to apply performance-driven refactoring techniques for managing spatial big data loads under uncertainty in the Cloud.

### 2.1 QoS Aware distributed batch spatial query processing and scalable storage

We have identified several problems in the traditional methods of big data partitioning and processing in Cloud deployments. Most importantly, general data partitioning does not play well with spatial data (e.g., spatial data locality preservation is normally neglected), leading to an antipattern that is analogous to *Circuitous Treasure Hunt*. This is so because not being able to preserve SDL means that spatial data are scattered in an inappropriate manner in the Cloud, thus requiring brute-force scans (broadcast operations versus targeted operations) that may span all computing nodes of the cluster. To avoid such antipatterns, we have designed QoS-aware spatial partitioning methods for scalable storage (through SpatialNoSQL[1] engineered atop MongoDB) and in-memory batch processing (through SpatialBPE incorporated within the layers of Apache Spark). One of the spatial partitioning methods that we have designed is an efficient STR-alike method, which resembles a rectangle packing algorithm. Other methods are based on dimensionality reduction (such as exploiting the Z-order curves and their applications to geohashes). This targeted refactoring design was successful in supporting alternative access patterns that avoid full scans. A partitioning method is a mean-to-an-end, where the main goal is improving the access performance for spatial analytics. We have also designed spatial-aware join processing algorithms [6] for NoSQL frameworks, which assists in answering advanced spatial analytics under uncertainty with QoS guarantees (lowering latency and maximizing resource utilization while keeping accuracy levels high).

Both SpatialNoSQL and SpatialBPE encompass models that continuously monitors recent execution status and apply specialized spatially-attuned self-adaptation models for managing the adverse effects of uncertainty, thereby applying convenient reconfigurations that impose minimal and negligible overheads which do not affect the general stability of the system.

### 2.2 Spatial Approximate Query Processing

SpatialNoSQL and SpatialBPE are just parts of the complete story. More interestingly in dynamic applications of smart cities and urban planning are spatial interactive analytics, where fast arriving fluctuating (i.e., in skewness and arrival rate) spatial data streams hit so hard the resources of the DSP engine. Those scenarios impose harsh requirements that do not appear to affect their static scenarios counterparts in the same manner. For example, stream spatial data processing systems should support incrementalization of spatial data stream computation, meaning that results are served incrementally based on time-based window semantics without the need to recompute or materialize previous loads. To achieve those goals, we have designed

[1] https://github.com/IsamAljawarneh/SpatialNoSQL

SpatialSPE [2], which aims at achieving low-latency and maximal resource utilization, while serving results with acceptable high accuracy expressed as rigorous error-bounds. SpatialSPE accepts a continuous query and QoS budgets (expressed as latency and accuracy targets) and employs our spatial-aware sampling method (that we dub as SAOS, which is an integral part of SpatialSPE) to select an appropriate spatially-representative sample, then it computes an approximate answer and serves it to the user incrementally, together with rigorous error bounds. Self-adaptivity in Software Performance Engineering (SPE) is a well-studied topic for traditional software systems. However, its application to Cloud-based big data management remains humble, especially for transient data with high fluctuation and oscillation in performance parameters (i.e., arrival and service rates). To close this void, I have designed SpatialSSJP to complement the end-to-end spatial-aware big data management in the Cloud. Specifically, SpatialSSJP is tightly coupled with SpatialSPE and supports its operations by introducing a self-adaptive control-theory based loop-feedback mechanism (specifically, PID controller, Proportional-Integrative-Derivative, , which acts as a latency-aware controller) in addition to another accuracy-aware controller (that is based on geo-statistics). All in all, enabling SpatialDSMS to survive during brutal burst spikes in data arrival rates by efficiently being able to tradeoff miniscule error-bounded accuracy for low-latency. SpatialSSJP is specifically designed to serve mixed-workloads in highly dynamic environments that require combining batch and streaming views (i.e., current views with historical views) or enriching spatial streams with static descriptions by employing a an efficient optimized spatial join (as opposed to a traditional resource-intensive brute-force counterpart).

## 3. Future research agenda

I have applied the algorithms that I have developed during my Ph.D. to the spatial dimension of the data. I aim, soon, to bring the time-series and temporal dimensions (probably also the contextual dimension) into the play. I believe that those dimensions are underrepresented when it comes to developing Cloud-based performance analysis processes. An obvious reason is that software architectural models are complex and performance problems can only be identified by being certain about the environment in which the system is designated to operate. For example, online interactive (i.e., dynamic) systems run into performance issues and complexities that do not often affect their static counterparts (i.e., systems running offline). Also, keeping track of all the performance metrics (i.e., indices) all the time can devalue the performance that is envisaged by parallel processing and storage. It can significantly increase the required storage space and response times, which is an undesirable antipattern in Cloud computing environments.

Accounting for spatial, temporal, and contextual data characteristics during the '*performance analysis'* phase in software performance engineering for Cloud-based frameworks is a challenging task. I have designed appropriate methods to achieve this for spatial dimension in the Cloud and I am currently working on extending those models so that they apply to the temporal dimension (including time-series analysis). This includes designing efficient and scalable algorithms for indexing, caching, and partitioning (for parallel execution in the Cloud) of temporal (also geo-referenced) data in addition to novel spatiotemporal query processing algorithms. Those algorithms will be backed by robust theoretical foundations such as spatial and temporal relational algebra. To my knowledge, there is no single framework that build on top of those which supports the synergy between spatial and temporal dimensions altogether. For example, being able to slice data by time

[2] The source code of SpatialSPE (together with SAOS sampling method) is available at: https://github.com/IsamAljawarneh/SpatialSPE

and location together has a weak support in the current literature. For instance, consider a query that asks about “how many rides originated in different neighborhoods of a city in 60 minutes buckets”. Such a stateful aggregation query cannot be efficiently answered by current state-of-art big data Cloud-based systems. I plan to enrich SpatialDSMS with aggregation compute-intensive time-series queries such as “moving average”, “Holt-Winters’ seasonal method” and many others.

Another pivotal research question that I am interested to tackle is the following. Ignoring performance analysis until later stages in the lifecycle of a Cloud-based big data management framework is tricky. Taking a “fix-it-later” reactive approach may potentially render the system unresponsive in production at critical vital times. Is it better to adopt a proactive approach for performance management of the big data systems? There are no obvious answers currently as most predictive proactive models may cause harmful instability if their prediction statistics fail to capture the real future behavior of the Cloud-based big data system. I aim at tackling this problem by designing model-based architectures that considers the stability of the systems as a first-class citizen. My plan is to inject a model-based performance stages early in the lifecycle of the software development process (focusing on spatiotemporal systems), thereafter adapting the architectural model of the system so that it guarantees achieving the QoS goals envisaged by business users. Current state-of-art solutions are tackling the problem for general data workloads. To my knowledge, the literature lacks specific abstractions of software performance engineering for spatially- and temporally-laden smart city scenarios. I argue that a full-fledged library should be designed for spatiotemporal big data management in the Cloud that considers, most importantly, detecting and solving Cloud-related antipatterns. Other research issues I am tackling are the following. What are some of the best methods and architectural models to meet performance requirements in dynamic application scenarios with fluctuating performance parameters (data arrival and operator service rates)? Which architectural model should be selected for a QoS-aware data management of spatiotemporal data: Cloud-based, or Fog-Cloud or Edge-Cloud? Adding the IoT Edge devices to the equation is another issue and has its complexities that normally do not affect Cloud-solo architectures.

I am a divergent thinker with deep hands-on experience in developing emerging solutions for complex problems that arise from dynamic applications scenarios in smart cities, urban planning, and mobility informatics by developing standard compliant prototypes tested on emerging Clouds (such as Amazon, and Microsoft Azure) in addition to on-premises computing clusters.

## REFERENCES


[1] N. Marz and J. Warren, *Big Data: Principles and Best Practices of Scalable Real-Time Data Systems.* New York; Manning Publications Co., 2015.

[2] ***I. M. H. Al Jawarneh***, "Quality of Service Aware Data Stream Processing for Highly Dynamic and Scalable Applications”, Ph.D. dissertation, Alma Mater Studiorum - Università di Bologna, 2020. Available: http://amsdottorato.unibo.it/9402/. **DOI** 10.6092/unibo/amsdottorato/9402.

[3] ***I. M. Al Jawarneh***, P. Bellavista, A. Corradi, L. Foschini and R. Montanari, "Locality-preserving spatial partitioning for geo big data analytics in main memory frameworks," in *Submitted to a Conference,2020*.

[4] ***I. M. Aljawarneh***, P. Bellavista, A. Corradi, R. Montanari, L. Foschini and A. Zanotti, "Efficient spark-based framework for big geospatial data query processing and analysis," in *2017 IEEE Symposium on Computers and Communications (ISCC),* 2017, pp. 851-856.

[5] ***I. M. Al Jawarneh***, P. Bellavista, A. Corradi, L. Foschini, R. Montanari and A. Zanotti, "In-memory spatial-aware framework for processing proximity-alike queries in big spatial data," in *2018 IEEE 23rd International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD),* 2018, pp. 1-6.

[6] ***I. M. Al Jawarneh***, P. Bellavista, A. Corradi, L. Foschini and R. Montanari, "Efficient QoS-Aware Spatial Join Processing for Scalable NoSQL Storage Frameworks," *Submitted to an IEEE Transactions Journal,* 2020.

[7] ***I. M. Al Jawarneh***, P. Bellavista, F. Casimiro, A. Corradi and L. Foschini, "Cost-effective strategies for provisioning NoSQL storage services in support for industry 4.0," in *2018 IEEE Symposium on Computers and Communications (ISCC),* 2018, pp. 1227.
[8] ***I. M. Al Jawarneh***, P. Bellavista, L. Foschini and R. Montanari, "Spatial-aware approximate big data stream processing," in *2019 IEEE Global Communications Conference (GLOBECOM),* 2019, pp. 1-6.
[9] ***I. M. Al Jawarneh***, P. Bellavista, A. Corradi, L. Foschini and R. Montanari, "Spatially representative online big data sampling for smart cities," in *Submitted to Conference,* 2020, .
[10] ***I. M. Al Jawarneh***, P. Bellavista, A. Corradi, L. Foschini and R. Montanari, "SpatialSSJP: QoS-Aware Adaptive Approximate Stream-Static Spatial Join Processor," *Manuscript,* 2020.
[11] ***I. M. Al Jawarneh***, P. Bellavista, A. Corradi, L. Foschini and R. Montanari, "Big Spatial Data Management for the Internet of Things: A Survey," *Journal of Network and Systems Management. Accepted (to Appear),* pp. DOI: 10.1007/s10922-6, 2020.