2021 IEEE Seventh International Conference on Big Data Computing Service and Applications (BigDataService) | 978-1-6654-3483-6/21/$31.00 ©2021 IEEE | DOI: 10.1109/BigDataService52369.2021.00022

Simulation Testbed for Evaluating Distributed Querying and Searching of Mass Spectrometry Big Data in a Network-based Infrastructure

Umair Mohammad and Fahad Saeed
School of Computing and Information Sciences, Florida International University
Miami, FL 33199, USA
Emails: {umohamma,fsaeed}@fiu.edu

***Abstract*—Advance access and reuse mechanisms for large-scale Mass Spectrometry (MS) data are essential for democratizing data for the omics research community and making it adhere to FAIR (Findable, Accessible, Interoperable, Reusable) principles. Although a number of centralized data repositories have been established, they have been limited to search mechanisms that depend on the meta-data associated with these MS datasets. Furthermore, they require constant influx of resources for maintenance. In this paper, we proposed an alternative novel distributed infrastructure for direct MS/MS spectral search. We designed and developed a simulation testbed using concepts from computer networks, queuing theory, and stochastic simulation methods. Results show that a distributed MS search based on raw MS/MS spectra can scale gracefully for up-to 2000 participating nodes, while simultaneously processing queries using the proposed networked infrastructure on the order of milliseconds to a few seconds for up-to a total of fifty billion MS/MS spectra.**

I. Introduction

The study of proteins is known as proteomics [1] and it is vital to study the structure of proteins and their contributions to biological processes because they are the building blocks of life. For example, proteomics is essential for understanding disease, designing customized drug delivery [2], and developing vaccines [3]; as witnessed during the COVID-19 pandemic. One of the most widely used instrument for characterizing proteins is the mass-spectrometer (mass-spec), which has enabled the era of shotgun proteomics. Modern high-throughput mass-spectrometry (MS) produces large amounts of data and thus, biologists have teamed up with computational scientists to facilitate rapid accurate analysis via computational methods.

Proteins are made up of amino acids that form peptide ion chains known as polypeptides. MS-based proteomics in a lab involves the preparation of a sample by digesting it in an enzyme such as trypsin. This fragments the proteins into different peptide ions. Next, the fragmented peptide ions are scanned in the mass-spec. This first scan results in a raw spectrum which plots the mass-to-charge (m/z) ratio against the discrete intensity/frequency peaks for each ion. The fragmented ions pass through the second scan to generate what is know as the raw MS/MS (or MS^2) spectra [1]. With the wide-spread availability of mass-specs and advent of high performance computing (HPC) coupled with large centralized protein databases such as PRIDE [4] and Uniprot [5], computational proteomics is evolving rapidly with several research labs across the world involved in these efforts.

Being able to deduce the peptides from raw experimental spectra is one of the most important steps in protein sequencing and mapping. Two search techniques used to achieve this goal are spectral library search and database search. Because the latter approach requires in silico digestion and generating model/theoretical spectra, it can end up taking up-to hours of execution time and terabytes of memory which makes it more suitable for offline analysis. On the other hand, spectral library search can be done in real time as long a database of well-known and defined theoretical/model spectra exists.

The current pandemic has highlighted the importance of collaborated research efforts for protein analysis. Although centralized repositories [4]–[6] facilitate collaboration and data sharing in proteomics, they involve searching with metadata and downloading huge data volumes, followed by database search [1]. Though useful, a more beneficial product for researchers will be a search engine that can accept raw MS/MS spectra as an input and provide matches to well-known and sequenced peptides. PeptideAtlas [7] is a tool that allows the user to perform such queries. This centralized repository only comprises 13,000 spectra whereas centralized repositories such as ProteomeXChange [6] contain billions of spectra. Scoring thousands of queries against billions of spectra is not scalable because of the linear time complexity of spectra matching algorithms. In contrast, parallelizing the spectral library search by splitting the model database across multiple nodes can enhance the scalability because the reduction in linear complexity computations dominates the communication cost of modern day servers.

Although distributing proteomics search with grid computing [8] and cluster computing [9] has been proposed before, the search mechanisms are based on metadata. In this paper, we propose a distributed framework which will allow various research labs and data centers to offer their stored theoretical MS/MS data for searching against raw MS/MS spectral queries. More specifically, we will concentrate on demonstrating that the distributed spectral library search system for proteomics will be scalable and outperform centralized approaches by designing a simulation testbed. This contribution is vital because it will help evaluate the scalability

978-1-6654-3483-6/21/$31.00 ©2021 IEEE
DOI 10.1109/BigDataService52369.2021.00022

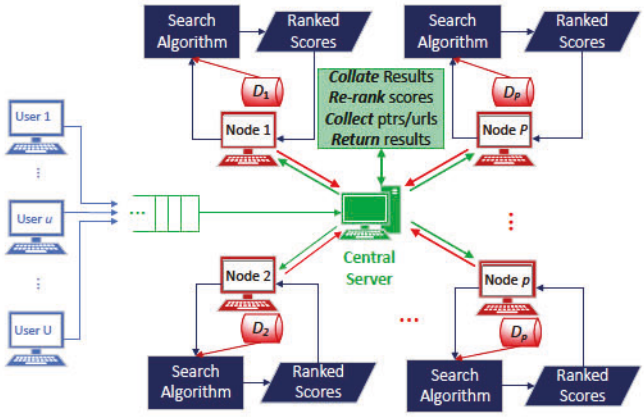

Fig. 1. Illustration of the online querying process with U users and P compute/storage nodes.

and test some of the fundamental limits with respect to varying loads, number of participants, and database sizes.

II. Distributed Database Design

In the context of proteomics related data, each lab equipped with MS equipment and regular workstations will analyze their own specimens and store the post-processed theoretical spectra locally. These stations will serve as the compute/storage nodes. A central server or data manager will keep a record of all spectral datasets at each node. Efficient indexing techniques have been presented for such MS/MS data and they can be used for this purpose [10]. Because this process is done offline at the time of data curation, it will not be counted towards the search time.

Whenever a user represented either by another compute/storage node in a participating lab or an external entity wants to run a spectral search, they will enter a query into a web-based form. The query will comprise the raw MS/MS spectra file and some other parameters such as the enzyme and precursor mass. The central server will perform basic preprocessing and forward the query spectra to all other nodes. Each node will run a search algorithm against each spectra in each of its local MS/MS dataset and generate a few matches ranked locally. Then, they will each send the results back to the central server in the form of scored URLs. The server will globally re-rank the scores and send a collection of results back to the user. Figure 1 illustrates this process.

A. Distributed Querying System Model

Consider a set of users $\mathcal{U} = \{1, \ldots, u, \ldots, U\}$ where each user $u \in \mathcal{U}$ can generate a query given by Q_u. Q_u may contain more than one raw spectrum up-to a certain limit. We assume that there exists a central server $\mathcal{S}$ and a set of compute/storage nodes $\mathcal{P} = \{1, \ldots, p, \ldots, P\}$ willing to participate. Each node $p \in \mathcal{P}$ has N_p $\forall p$ spectral datasets and the size of the datasets is load-balanced across P nodes. Basically, each MS/MS dataset contains $D_{p,i}$ spectra itself where $i = 1, \ldots, N_p$ $\forall p$.

The central server itself is connected to the nodes using long-haul broadband links (these may be fully wired using a combination of Ethernet and fiber). The user-server links may also include wireless communication. To model these links, we will utilize previous results to model the channel rate. For example, every user can communicate to the server with an uplink rate of $R_{u,D}$ and the server with the downlink rate of $R_{u,D}$ bits/sec. Similarly, each node p can communicate to the server with uplink rate $R_{p,U}$ and downlink rate $R_{p,D}$ bits/sec. The server is assumed to have a computational capability of f_S Hz and each node has a capacity of f_p Hz.

Any user can send a query of size $\mathcal{L}_{Q,u}$ bytes to the server. Furthermore, we can assume that a uniform storage system exists such that each raw MS/MS spectra is stored using $\mathcal{L}_D$ bytes across all nodes, the user submitted query spectrum is standardized by the server into $\mathcal{L}_Q$ bytes, and each result (pointer/URL/etc) generated by node p is stored with $\mathcal{L}_R$ bytes. However, in response to a query, each node will generate M_p $\forall p$ results or matches. The final result the server shares with the user u after collating all the results can be of size $\mathcal{L}_{R,u}$ bytes.

After the server has received the query spectra from any user, the distributed system will undergo the following steps. The server will transmit the query spectra to each compute/storage node with time $t_{p,D}$ $\forall p \in \mathcal{P}$. Each node $p \in \mathcal{P}$ will also consume time $t_{p,E}$ to run the search algorithm against every entry in its database. Lastly, each node will send M_p results back to the central server with time $t_{p,U}$ $\forall p \in \mathcal{P}$. Equations (1)-(3) represent the above described downlink, execution, and uplink times, respectively, for each node $p \in \mathcal{P}$.

$$t_{p,D} = \frac{8\mathcal{L}_Q}{R_{p,D}}, \ \forall p \in \mathcal{P} \tag{1}$$

$$t_{p,E} = \frac{f_r}{f_p}(mN + c) \qquad \forall p \in \mathcal{P} \tag{2}$$

$$t_{p,U} = \frac{8\mathcal{L}_R M_p}{R_{p,U}}, \ \forall p \in \mathcal{P} \tag{3}$$

In general, the time consumed by node p will be a function of the ratio of its computational capacity f_p $\forall p$ to the capacity of the reference machine f_r on which the spectral search algorithm was evaluated and the execution time in seconds. Typically, the execution time for most spectral search algorithms is of linear time complexity with respect to the number of spectra N where m and c are the slope and intercept of the line, respectively.

From the server's perspective, the total time needed by one compute/storage node t_p $\forall p \in \mathcal{P}$ to return back the search results can be described by:

$$t_p = t_{p,D} + t_{p,E} + t_{p,U} \tag{4}$$

As the server is limited by the worst performing node, the actual time t_S needed by the server to receive the results back from all nodes can be given by:

$$t_S = \max\{t_p\} \tag{5}$$

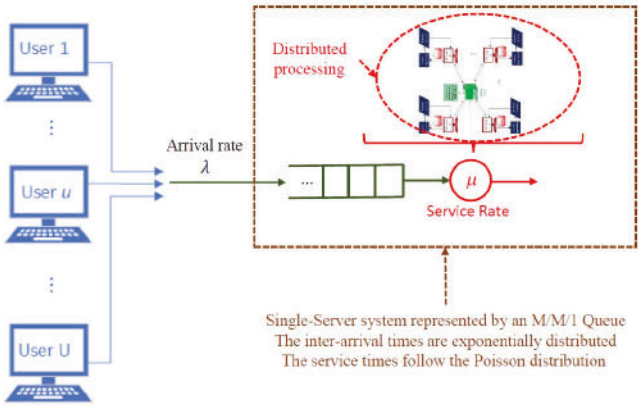

Fig. 2. Illustration of single-server model represented by an $M/M/1$ queue. The service rate μ encompasses the distributed query search process.

III. Simulation Strategy

Once the user submits a query, the time taken for them to get back the result will include the process of sending a query spectra to the server, time spent waiting for the server to complete the distributed processing described in section 2, and acquiring the results (comprising ranked URLs/pointers). The distributed query spectra dissemination and search algorithm time dominates the communication time between the server and user due to the linear time complexity of spectral matching algorithms. Furthermore, the communication time between any user and the server will impact all users differently according to the strength of their personal networks whereas the distributed query processing time will impact all users equally. Therefore, we focus on modeling the impact of various system parameters on the completion time of the distributed query processing section.

One way to model a system where several users submit requests that are served by one or multiple servers is a queue. In our case, because we are focused on distributed query processing, we can model this process as a single server operation. This simple simulation environment can be modeled by an $M/M/1$ queue as shown in Figure 2. This implies that the arrival of queries is assumed to follow the Exponential distribution with an average rate λ and the server processing follows the Poisson process with average service rate μ. The mean service time $x = 1/\mu$. The service time will encapsulate the distributed search algorithm processing described in Figure 1. Therefore, we can pre-evaluate an average service time $x \leftarrow t_S^{avg}$ of the overall system to one query by running Monte Carlo simulations using stochastic models to emulate the communication links and computational capacities of each node. For an experiment with a total of J runs, and the maximum service time at run $\#j$ denoted by $t_S^j \ \forall \ j \in \{1, \ \ldots, \ J\}$, the mean service time can be given by:

$$t_S^{avg} = \frac{1}{J} \sum_{j=1}^{J} t_S^j \tag{6}$$

We run this simulation for different settings by varying the numerical values of the parameters such as the number of spectra and the comp/comm capabilities of each compute/storage node, and determine the expected average service time x for each scenario. Previously, physical parameters such as per node processor capability f_p, uplink rate $R_{p,U}$, and downlink rate R_{p_D}, have been modeled from a uniform random distribution $\mathcal{U}[A, B]$ where A and B are real numbers representing the start and end of the possible range of values of the variable. Further details are given in the next sub-section.

Although this approach does not allow for the modeling of packet arrivals, for query response time modeling, a different approach has been successfully used previously [11]. Given a mean service time of a server x, the service time $\bar{x}$ in the presence of N_q concurrent requests can be modeled as a degradation of the original response time x by a load factor p as shown in (7).

$$\bar{x} = x(1 - \rho)^{N_q} \tag{7}$$

A. Simulation Parameters

We assume that the there are a total of P nodes that own a total of N spectral MS/MS datasets such that the data is equally split among the nodes with each node p comprising $N_p = N/P$ spectral datasets. We model the number of spectra per dataset by a uniform distribution such that $D_{p,i} \sim \mathcal{U}[400, 600] \times 10^3$, $i = 1, \ldots, N_p \ \forall p$. Because there are equal datasets per node, each node ends up with similar number of spectra on average and hence, the system is load-balanced.

Furthermore, modern servers can support downlink speeds of up-to gigabits per second (Gbps) for multiple clients at once. Because we are modeling large values of P and nodes may comprise a combination of wireless and wired links, we will assume uplink speeds of up-to hundreds of megabits per second (Mbps). (In future works, we will model the communication links as separate queues to more accurately capture the behavior of the channels.) We will model $R_{p,U} \sim \mathcal{U}[80, 100] \times 10^6$ and $R_{p,D} \sim \mathcal{U}[100, 300] \times 10^6$ bits per second. Moreover, it is assumed that a single query spectra will be uploaded as a packet of size $\mathcal{L}_Q = 32$ kilobytes by the server whereas the results are assumed to be stored as packets of $\mathcal{L}_R = 1024$ bytes $\ \forall p$. We assume that each node will generate results in the range $M_p \sim \mathcal{U}[0, 150] \ \forall p$. We will assume that the server applies certain preprocessing steps which will reduce the number of datasets needed for comparison. We will simulate this by choosing P' out of P nodes for each request. The set of $P - P'$ dropped nodes will be generated randomly by ignoring around 20% of the nodes in each cycle. The corresponding datasets will also be dropped when simulating the centralized case.

It is assumed that the processing capability of each node varies from $f_p \sim \mathcal{U}[2, 3] \times 10^9$ Hz $\ \forall p$. The search algorithm employed is the SpectraST [12] which uses the underlying algorithm described in [13]. The time taken per query spectrum is 0.005s to run a search against $N = 30000$ spectra. Assuming

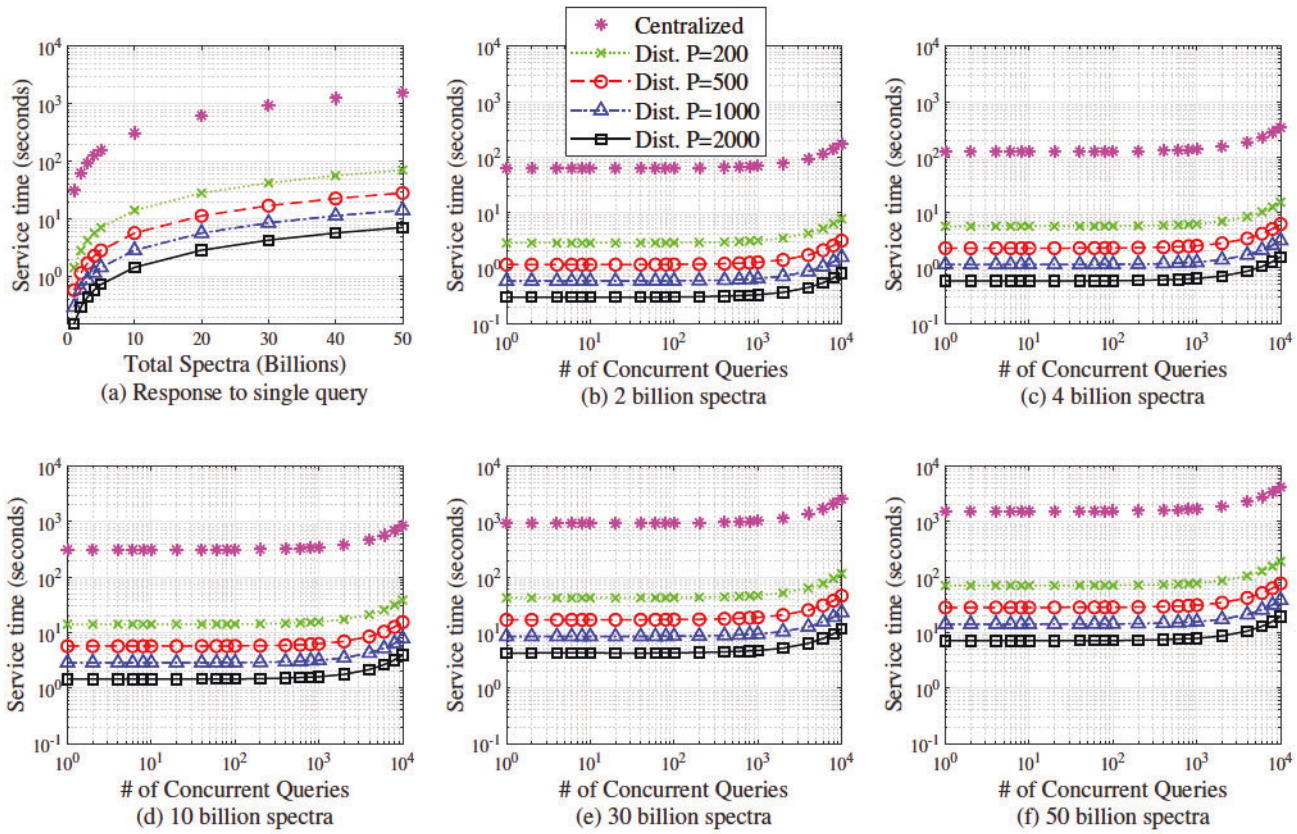

Fig. 3. Response time of distributed querying across 200, 500, 1000, and 2000 node systems to (a) a single query spectra searched against up-to 50 billion model spectra (b)-(f) up-to 10k concurrent requests for a system comprising a total of N = {2, 4, 10, 30, 50} billion theoretical spectra, respectively.

a starting point of $(0, 0)$, we can set the slope $m = 0.005 * 30e-3$ and $c = 0$. These evaluation results are for an $f_r = 3.4$ GHz machine. The load factor is set to $\rho = 0.001\%$ initially based on the capability of modern servers of handling up-to 10,000 simultaneous connections. We also test the system for several values of the load factor ρ from the range of $1e-6$ to 1. In this case, the inability of the system to scale at all is represented by 1 and increases as the load factor reduces.

Lastly, to simulate the performance of the central server-node communication with increasing load, we design a load factor that takes into account the computation and communication separately such that $p = 0.5\rho_{comp} + 0.5\rho_{comm}$. Because the query preprocessing and forwarding requires minimal computation, ρ_{comp} is set to a constant value of 1e-4. On the other hand, modern day servers can handle hundreds of Gigabit connections without degradation for hundreds of channels. Beyond that, the performance suffers due to physical layer limitations. Therefore, ρ_{comm} is set as constant for up-to 500 nodes and then becomes inversely proportional to P. Table I describes the parameters used for the simulation of the system.

TABLE I
LIST OF SIMULATION PARAMETERS.

Parameter	Value/Range
P	$\{200, 500, 1000, 2000\}$
N	$1 - 50$ billion spectra
N_q	$1 - 10,000$ concurrent queries
f_r	3.4 GHz
f_p	$\sim \mathcal{U}[2, 3] \times 10^9$ Hz
$R_{p,U}$	$\sim \mathcal{U}[800, 1000]$ Mbps
$R_{p,D}$	$\sim \mathcal{U}[80, 100]$ Mbps
$D_{p,i}$	$\sim \mathcal{U}[4, 6] \times 10^4$
M_p	$\sim \mathcal{U}[0, 150]$
$\mathcal{L}_Q$	32*1024 bytes
$\mathcal{L}_R$	512 bytes

IV. Results and Discussion

Figure 3 illustrates the service times of the distributed system to a single query and N_q concurrent query spectra. We tested for the values of $P \in \{200, 500, 1000, 2000\}$ compute/storage nodes for total spectra in the range 1 billion to 50 billion. As the plots clearly demonstrate, the distributed querying systems significantly outperform the centralized scheme. Figure 3a shows that as we increase the number of participating nodes, the response time to a single query reduces. For example, a system with 200 nodes can process 10 billion spectra in 30s whereas the centralized system will consume 600s, a twenty-fold increase. Furthermore, increasing the number of participants to 2000 nodes will result in an

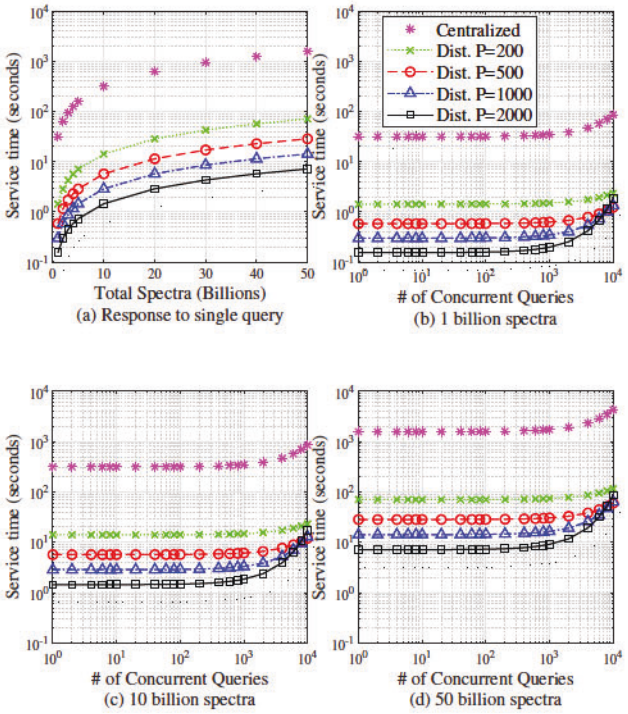

Fig. 4. Response time of distributed querying across 200, 500, 1000, and 2000 node systems to (a) a single query spectra searched against up-to 50 billion model spectra (b)-(d) up-to 10,000 concurrent requests for a system comprising a total of N = {1, 10, 50} billion theoretical spectra, respectively.

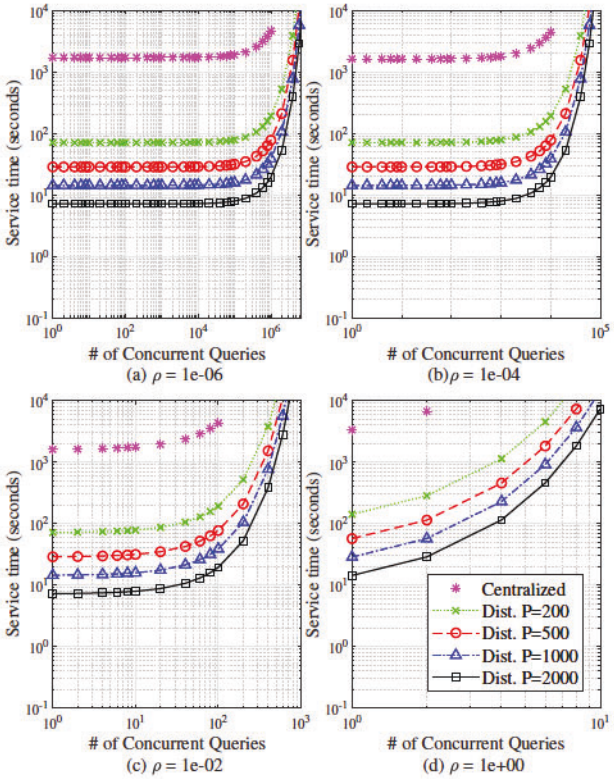

Fig. 5. Response times to concurrent queries with variable load factors.

execution time of 2s, just 6.67% of the execution time needed for 200 nodes and 0.3% compared to the centralized system.

Subsequent plots in figures 3 (b)-(f) demonstrate the ability of the distributed system to scale for up-to 10000 concurrent query spectra for different search database sizes ranging from 2 to 50 billion spectra. For example, a system of more than 500 nodes can provide constant time performance in the range of a few milliseconds to seconds for up-to 1000 concurrent queries for a search against 2 billion spectra. A system with 2000 participating nodes can provide service times of 0.5, 1, and 7 seconds for searches against 2, 10, and 50 billion spectra, respectively, for almost 10,000 concurrent queries.

Figure 4 illustrates the impact of the degradation of server-node communication due to a high number of links. Recall that the load factor was made a function of the computation (constant at 0.001%) and communication (inversely proportional to P). It is only when the number of concurrent requests increase from 1k to 10k that the performance degrades exponentially for larger numbers of nodes. In all cases, the distributed systems still perform better. For example, a 2000 node system provides lower service times compared to 1000 node systems for up-to 1k concurrent queries. In contrast, with 10k concurrent requests, 1000 nodes provide a slightly better performance because of a reduction in scalability. However, as all schemes start degrading exponentially at those levels of load, we can say that increasing the number of nodes provides a reduced response time for up-to a high range of concurrent requests. In a real-system, even at high load times the requests that are serviced will leave. Therefore, it is reasonable to expect that less than ten thousand concurrent query spectra will be present in the system.

A. *Distributed System Resilience and Solutions*

Figure 5 demonstrates the resilience of the distributed system as opposed to the centralized scheme in responding to multiple concurrent queries for searching against 50 billion MS/MS spectra under variable load factors. The value of $\rho = 1$ represents the worst case scenario, where a single additional query doubles the system workload. For a fair comparison, we first show that given same levels of degradation for both centralized and distributed schemes, the distributed schemes perform better and in general, provide a lower service time in the constant region. For example, when $\rho = 1e-4$, the service time increases exponentially and approaches the centralized service time only when the system has 100,000 concurrent requests. Because the number of query spectra per is are limited, this scenario will only occur if there is a very well co-coordinated cyber-attack. Simple mitigation technique such as captchas can be used for users sending excessive requests in addition to other security measures for complex attacks.

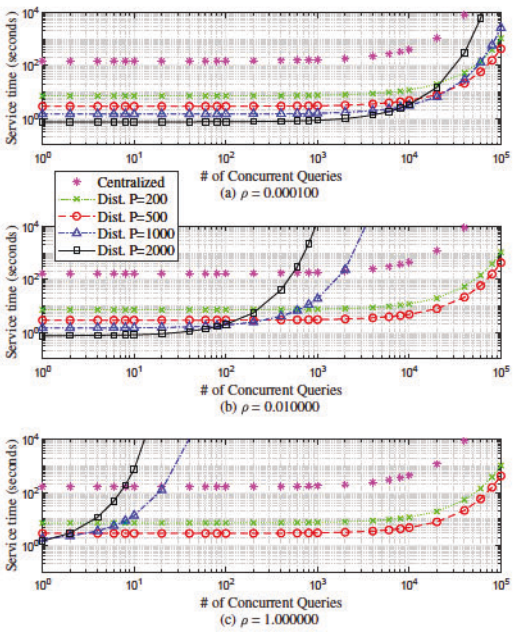

Fig. 6. Response time to concurrent queries with communication degradation.

Figure 6 further studies the resilience of the system under communication link degradation only. In this scenario, we keep the impact of multiple queries on the performance of the centralized system and the computational part of the distributed system at 0.001%. In contrast, the communication degradation is represented by $\rho_{comm} = 1e-4$, $1e-2$, and 1. Because the system can handle seamless parallel communication for up-to $P = 500$, the performance does not degrade severely. On the other hand, for $P = 1000$ and $P = 2000$ the system degradation is more severe with exponential service times for a low number of concurrent requests. Typically, it is expected that the degradation will not reach the levels represented by $\rho_{comm} = 1$ because that effectively means the central server can communicate with 1 node at a time only.

One effective solution is to design a smart data redundancy policy for MS/MS spectra stored at multiple locations. This will increase the overall memory requirements but improve system resilience. Furthermore, it will allow the central controller to choose the nodes with the best links or drop nodes where the communication capabilities fall below a pre-defined threshold. Moreover, the nodes can also be dropped according to the volume of concurrent requests where for lower N_q. For example, for lower N_q values, $P = 1000$ still performs better than $P = 200$ or 500. Overall, a distributed system with a large number of nodes reduces service times by 10 to 100-folds. The high scalability and the benefits for spectral search makes it worthwhile to explore the design and implementation of a distributed data access infrastructure for big proteomics data based on paradigms such as volunteer computing.

V. Conclusion

In this work, we have designed a simulation testbed to show that a distributed system for proteomics data search can scale and provide improved performance over centralized databases. Moreover, it was shown that even in band-limited channels, the system would scale well for very large numbers of concurrent query spectra. Despite these promising results, several limitations need to be addressed to complete the simulation testbed including modeling the communication links as separate channels, incorporating node selection strategy based on real-time physical capabilities, and a cost-based model to reward participating labs better for sharing their resources and data with the proteomics community.

References

[1] J. Colinge and K. L. Bennett, "Introduction to Computational Proteomics," *PLoS Computational Biology*, vol. 3, no. 7, p. e114, jul 2007. [Online]. Available: https://dx.plos.org/10.1371/journal.pcbi.0030114

[2] A. Vedadghavami, C. Zhang, and A. G. Bajpayee, "Overcoming negatively charged tissue barriers: Drug delivery using cationic peptides and proteins," p. 100898, oct 2020.

[3] A. Bolhassani and S. Rafati, "Heat-shock proteins as powerful weapons in vaccine development," *Expert Review of Vaccines*, vol. 7, no. 8, pp. 1185–1199, oct 2008. [Online]. Available: https://www.tandfonline.com/doi/abs/10.1586/14760584.7.8.1185

[4] F. Reisinger, N. Del-Toro, T. Ternent, H. Hermjakob, and J. A. Vizcaino, "Introducing the PRIDE Archive RESTful web services," *Nucleic acids research*, vol. 43, no. W1, pp. W599—W604, 2015.

[5] U. Consortium, "The universal protein resource (UniProt)," *Nucleic acids research*, vol. 36, no. suppl_1, pp. D190—-D195, 2007.

[6] J. A. Vizcaino, E. W. Deutsch, R. Wang, A. Csordas, F. Reisinger, D. Rios, J. A. Dianes, Z. Sun, T. Farrah, N. Bandeira, and Others, "ProteomeXchange provides globally coordinated proteomics data submission and dissemination," *Nature biotechnology*, vol. 32, no. 3, p. 223, 2014.

[7] J. M. Schwenk, G. S. Omenn, Z. Sun, D. S. Campbell, M. S. Baker, C. M. Overall, R. Aebersold, R. L. Moritz, and E. W. Deutsch, "The Human Plasma Proteome Draft of 2017: Building on the Human Plasma PeptideAtlas from Mass Spectrometry and Complementary Assays," pp. 4299–4310, dec 2017. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/28938075/

[8] P. Veltri, M. Cannataro, and G. Tradigo, "Sharing mass spectrometry data in a grid-based distributed proteomics laboratory," *Information Processing and Management*, vol. 43, no. 3, pp. 577–591, may 2007.

[9] K. Verheggen, H. Barsnes, and L. Martens, "Distributed computing and data storage in proteomics: Many hands make light work, and a stronger memory," *Proteomics and Systems Biology*, vol. 14, no. 4-5, pp. 367–377, mar 2014. [Online]. Available: http://doi.wiley.com/10.1002/pmic.201300288

[10] M. Haseeb and F. Saeed, "Efficient Shared Peak Counting in Database Peptide Search Using Compact Data Structure for Fragment-Ion Index," in *2019 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*. IEEE, 2019, pp. 275–278.

[11] M. Kihl, G. Cedersjö, A. Robertsson, and B. Aspernäs, "Performance measurements and modeling of database servers," in *European Conference on Service-Oriented and Cloud Computing*, 2012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1748013220300670

[12] H. Lam, E. W. Deutsch, J. S. Eddes, J. K. Eng, N. King, S. E. Stein, and R. Aebersold, "Development and validation of a spectral library searching method for peptide identification from MS/MS," *Proteomics and Systems Biology*, vol. 7, no. 5, pp. 655–667, mar 2007. [Online]. Available: http://doi.wiley.com/10.1002/pmic.200600625

[13] H. Lam, E. W. Deutsch, J. S. Eddes, J. K. Eng, S. E. Stein, and R. Aebersold, "Building consensus spectral libraries for peptide identification in proteomics," *Nature Methods*, vol. 5, no. 10, pp. 873–875, sep 2008. [Online]. Available: http://npg.nature.com/reprintsandpermissions/