A Visual Analytics Paradigm Enabling Trillion-Edge Graph Exploration

Pak Chung Wong[1,*], David Haglin[1], David Gillen[1], Daniel Chavarria[1], Vito Castellana[1], Cliff Joslyn[1], Alan Chappell[1], and Song Zhang[2]

[1]Pacific Northwest National Laboratory, [2]Mississippi State University

ABSTRACT

We present a visual analytics paradigm and a system prototype for exploring web-scale graphs. A web-scale graph is described as a graph with ~one trillion edges and ~50 billion vertices. While there is an aggressive R&D effort in processing and exploring web-scale graphs among internet vendors such as Facebook and Google, visualizing a graph of that scale still remains an underexplored R&D area. The paper describes a nontraditional *peek-and-filter* strategy that facilitates the exploration of a graph database of unprecedented size for visualization and analytics. We demonstrate that our system prototype can 1) preprocess a graph with ~25 billion edges in less than two hours and 2) support database query and interactive visualization on the processed graph database afterward. Based on our computational performance results, we argue that we most likely will achieve the one trillion edge mark (a computational performance improvement of 40 times) for graph visual analytics in the near future.

Keywords: Visual analytics, graph visualization, web-scale graph, NetFlow, triple-store, RDF, SPARQL, high-performance computing, Linux cluster

1 INTRODUCTION

In today's big data era, a *web-scale* graph is often described as a graph with approximately one trillion edges and approximately 50 billion vertices [4][5]. Exploring a web-scale graph is currently a hot topic in cloud-computing and cyber-analytics communities led by major internet vendors such as Google, Facebook, Microsoft, and Amazon. Although cutting-edge graph exploration technologies, such as Giraph [2] and Pregel [19], have been reported in the literature, *none* use visualization in their analytical solution. This paper investigates the web-scale graph exploration problem from a new visual analytics perspective.

Unlike exascale scientific data that could potentially be analyzed through *in situ* visualization [6] on the same machine that models the data, web-scale graphs are often harvested from outside the visualization machine and thus create a laundry list of big-data problems in *volume*, *velocity*, and *variety*. Ingesting the data alone will become a daunting hurdle to overcome because a web-scale graph will occupy 29.1TB when stored as an edge list, 29.5TB as an adjacency list, and 271EB (exabyte) as an adjacency matrix [4].

We propose to work around the size problem by creating a new *peek-and-filter* visual analytics paradigm, which provides a glimpse of the underlying data and allows users to decide if they want to commit to a potentially costly visualization step. We further develop a visual analytics prototype that applies the peek-and-filter strategy twice, separately on both high-performance computing (HPC) and desktop levels to explore benchmark graphs. Preliminary results indicate that the system prototype can 1) preprocess (i.e., data ingest and index creation) benchmark graphs with ~25 billion edges in less than two hours and 2) support database query and visualization on the processed graph database afterward. Compatible technologies reported in the literature would have required days to just ingest graphs of similar sizes [10].

The paper describes the proposed paradigm and the system prototype that integrates a visualization frontend and a resource description framework (RDF)-based [23] graph database backend, evaluates computation performance results using benchmark datasets, shares user experiences and lessons learned, and discusses our ongoing effort to scale up from exploring 25 billion to one trillion edges.

* pak.wong@pnnl.gov

IEEE Symposium on Large Data Analysis and Visualization 2015
October 25–30, Chicago, Il, USA
978-1-4673-8517-6/15/$31.00 ©2015 IEEE

2 RELATED WORK

The graph analytics literature is rich and abundant, but only a few focus on web-scale graphs. Burkhardt and Waring [4][5] give an inspiring discussion of web-scale graph problems and challenges on topics such as exploration and analytics, algorithm, hardware, and storage in a real-world setting. The term "web-scale" as described in Burkhardt and Waring [4][5] is sometimes referred to as "Google-scale" or "Facebook-scale" in the literature.

We organize the rest of the related-work discussion into two categories: graph visual analytics and web-scale graph processing. Although the former does not directly address web-scale graphs and the latter does not involve visualization development, together they inspired and informed the R&D of the peek-and-filter paradigm discussed in this paper.

2.1 Big Graph Visual Analytics

We agree that the graph size that seems big today is different from what seemed big only a few years ago. Here we highlight a few notable works that address different aspects of big graph visual analytics from exploration, to layout, to user interaction.

Rohrer et al. [24] give a thorough review of big-graph visual analytics through practical examples. The paper discusses the research challenges of big data analytics, describes two working visual analytics tools in social network and text repository visualization, and highlights the visual analytics contests at the annual IEEE VAST Conference.

Instead of machine-automated or user-assisted visualization of big graphs, Yuan et al. [33] apply a crowdsourcing approach to develop an *intelligent graph layout* that merges many subgraphs submitted by a crowd. The big-graph layout approach, which seeks both symmetry and aesthetics in graph visualization, can be extended to analyze other crowdsource-based data.

Hadlak et al. [11] introduce an interesting concept of *in situ visualization*, which allows users to interactively switch to different visualization techniques (and thus the focus of analysis) during the course of big-graph exploration. Notice that the use of the term "in situ" here is different from that used in scientific visualization [6].

In big-graph visualization, the problems caused by drawing graph edges are far more difficult to address than those brought by drawing graph vertices. Xu et al. [32] conduct a user study on multiple prevailing techniques of drawing graph edges. The study highlights the strengths and weaknesses of different edge-drawing

techniques for different big graph layout preferences and requirements.

Big graph visual analytics is seldom supported by one single visual technique but instead employs a multifaceted array of techniques that addresses and captures multiple aspects of graph visual analytics in concert. Hadlak et al. [12] give the most complete survey study to date on *multifaceted graph visualization* categorized in four facets of partitions, attributes, time, and space. The paper cites more than a hundred examples in four categories of applications. Another thorough survey on big graph visual analytics is Von Landesberger et al. [28], which emphasizes individual visualization techniques instead of multifaceted techniques.

2.2 Web-Scale Graph Processing

In recent years, there has been a push toward supporting web-scale graph analytics. Such support is the goal of these software solutions: Pregel [18], Giraph [2][7][8], Mizan [16], GraphLab [17][18], and GPS [25].

All are graph-based in their data abstraction and expose an API that encourages programmers to "think like a vertex." This vertex-centric approach is built around a programming model where the user describes actions to be taken at each vertex of the graph given current inputs. The strategy works by major iterations of vertex calculations and terminates when all of the vertices have agreed that the computation is done. This programming model works well for algorithms such as PageRank.

Some specific specializations on the "think like a vertex" theme are Mizan and GraphLab. Mizan works to optimize performance based on load balancing and graph partitioning [15]. GraphLab focuses more on machine learning than general graph algorithms [16][17].

3 Graph Visual Analytics Challenges

We discuss four challenges facing web-scale graph visual analytics.

3.1 Size Matters

Much of today's big graph visualization R&D focuses on what we characterize as *micro exploration*, as illustrated in Figure 1. Part of the visual analytics process often happens at the individual vertex and edge levels, such as pinpointing a particular person or transaction in a social network through, for example, the inspection of a force-directed graph layout.

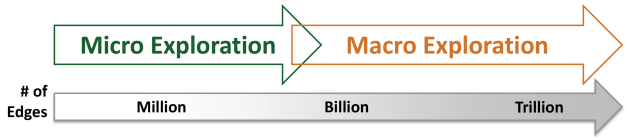

Figure 1: Data size ranges for micro and macro explorations.

When the underlying graph size reaches web-scale, we enter the largely unexplored territory of *macro exploration* in graph visual analytics. Graphs of this size will most likely be stored on and processed by an HPC machine. Everything from visualization, to computation, to user interaction designed for micro exploration will have to change to adapt. One visualization goal for macro exploration is to break down the web-scale graph to a size at which micro exploration can be conducted.

3.2 Cognitive Scalability

Most agree that there is no Moore's Law for cognitive activities. Human cognitive biases often inhibit understanding of graph structures. When a graph grows to thousands to tens of thousands of edges, structures within a visual layout will start to blend in like background white noise.

When the graph size reaches web-scale, the role of visualization will most likely be limited to showing the properties and characteristics of the graph instead of the graph itself.

3.3 Multilevel Visualization Scalability

Multilevel visualization is a proven strategy found in big-graph visualization literature [14][31]. Given a big dataset G, a multilevel visual analytics approach generates an increasingly coarsened hierarchy $G_0, G_1 \ldots G_i \ldots G_n$ such that the fine levels provide local details and the coarsened levels give the overall structures of G. The multilevel approach addresses many big graph visual analytics problems when the graph size is in the range of tens of thousands to hundreds of thousands of edges.

When the graph size reaches web-scale, the number of hierarchical levels quickly escalates. A trillion-edge graph will have as many as 40 levels within a dyadic hierarchy. Navigating such a deep hierarchy and searching for answers can itself be a real challenge.

3.4 Graph Complexity and Database

The fourth challenge is the complexity of the underlying data, which is precisely defined as an *attributed*, *directed*, *multi-graph*. The graph vertices have attributes. The graph edges have a direction and may have multiple attributes. There may be millions of edges with different directions and/or attribute values between any two graph vertices.

Using a relational database to manage such a complex graph will inevitably create many normalized tables that will need to be joined before a database query can be processed. The requirement of joining these tables (i.e., expanding the schema) will become increasingly costly and eventually prohibitive as the graph size continues to increase.

Our current system prototype uses RDF to address some of the above performance issues brought on by a relational database. See Sections 5.1 and 5.2 for further details on RDF.

4 Traditional and New Exploration Approaches

We argue that when exploring web-scale graphs, the role of visualization will be better off to move away from the traditional *drawing-centric* approach that surrounds the visual analytics with a graph depiction and move to a *peek-and-filter* approach that first surmises the cost and consequence of different queue options before deducing the scope of visual analytics.

Here we will use two graph analytics tools developed at the Pacific Northwest National Laboratory (PNNL), GreenHornet [31] and T.Rex [27], to illustrate the strategy differences between the two approaches. In Section 5, we will show that the impacts of the paradigm change go beyond the user interaction design and spread into the design of the analytics and computation infrastructure.

4.1 Drawing Centric

We first use GreenHornet [31] to demonstrate the concept of the drawing-centric exploration approach. Other prevailing graph analytics tools such as Gephi [9] and Renoir [22] also have a similar interaction design. Figure 2 describes one way to use GreenHornet to visually analyze a graph.

In step 1, a user brings up a drawing layout of the targeted graph and then adjusts the visualization resolution (drawing details) using a scroll widget in step 2. In step 3, the user browses the data using histograms on the right and a spreadsheet at the bottom. In step 4, the user can drill into the graph by aggregating the vertex/edge parameters through different selection means. Step 5 redraws the graph after parameter aggregation. The above steps will repeat until an answer is found or a theory is developed. The graph drawing remains the centerpiece of exploration throughout the process.

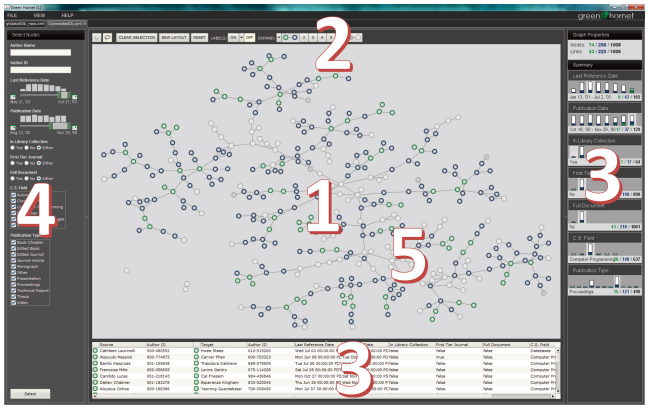

Figure 2: Drawing-centric visual analytics using GreenHornet.

4.2 Peek-and-Filter Centric

Figure 3 uses a series of T.Rex screenshot cut-offs to illustrate the concept of the proposed peek-and-filter approach. (We will describe the T.Rex system in detail in Section 6.3.) The example in Figure 3 uses an open source NetFlow dataset provided by the IEEE VAST Challenge 2013 [15].

In step 1 of Figure 3, we "peek" the data and find the data size (6,939,698 NetFlow records) too big for graph visualization. Step 2 shows a facets tool, which is a spreadsheet-like scrollable data table. Inside the first facet highlighted by a red rectangle under the source IP address (SIP) column, the SIP address 10.15.7.85 (upper left) has 1017458 (upper right) occurrences in the NetFlow dataset depicted temporally by the histogram (with green spikes) underneath the SIP address and counter within the red rectangle.

Assume we select the facet with SIP equal to 10.9.81.5 (i.e., the second facet under the SIP column), the corresponding facet in step 3 now turns green. All facets in the other columns are updated instantly. For example, the counter of destination IP (DIP) address 171.10.014 changes from 490657 in step 2 to 21478 in step 3 (as shown by the red arrow on the right). One of the spikes in the corresponding histogram (highlighted by a red circle) turns from green in step 2 to gray (i.e., filtered away) in step 3 after the SIP selection. The filtered graph size now has 291,323 NetFlow records as shown in step 4.

To complete the visual analytics cycle, the filtered data will be visualized within T.Rex, as described later in Sections 6 and 7.

4.3 Distinguish from the Conventional Wisdom

Shneiderman's visual information-seeking mantra, which states "*overview first, zoom and filter, then details-on-demand,*" has influenced the thinking of the entire information visualization community for nearly two decades. Our peek-and-filter concept represents a paradigm shift from conventional belief of using an overview visualization to guide data analysis to the new belief of first determining the right big-data mix before conducting visual analytics. The former is generally considered a top-down approach of visualizing the whole before the parts. The latter discourages visualizing the whole dataset at any point during the exploration.

5 Data Management, Databases, and Computation

Before introducing the system prototype in detail, we first describe the graph databases and hardware systems that power the prototype and explain the rationale behind our system choices. Bear in mind that the plug-and-play nature of our system design allows future modification to the infrastructure, such as replacing the graph database or the visualization for different analysis purposes.

5.1 Resource Description Framework (RDF)

It was determined early in the project that big graphs, as described in Section 3.4, would best be represented and stored as RDF [23] triples in the form of subject-predicate-object (s-p-o), e.g., Zuckerberg (subject)-is CEO of (predicate)-Facebook (object).

In an RDF-based system, the predicate of a triple is explicitly defined, which provides greater flexibility and scalability for database maintenance and query optimization. More importantly, the triple s-p-o design supports federalized (or decentralized) queries naturally and avoids costly schema expansion like the relational database queries.

Today's prevailing RDF data stores include Facebook's Open Graph and Google's RDFa. To power our system prototype, we implement a customized version of SPARQL [26]—an RDF query

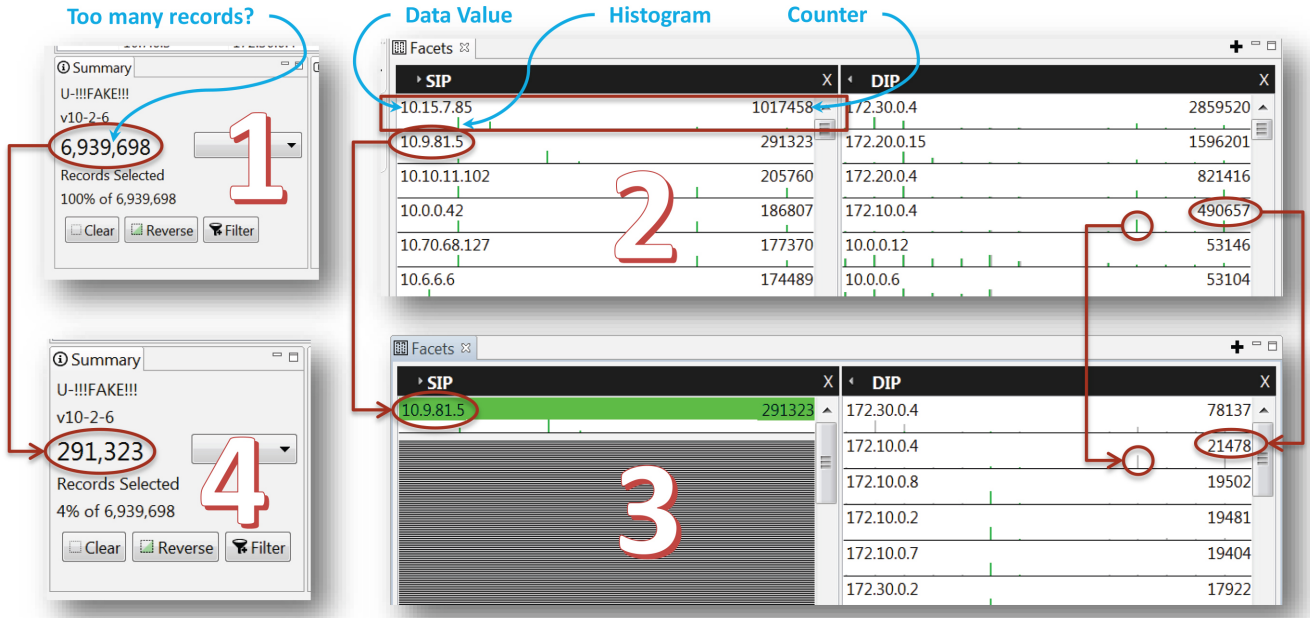

Figure 3: Peek-and-filter centric visual analytics using T.Rex.

language—in GEMS [10][20][29] (Section 5.2) running on a customized HPC machine (Section 5.3).

5.2 GEMS

GEMS [10][20][29] is developed at PNNL for exploring big graphs on HPC machines. The command-line-based system implements a subset of SPARQL query language with customized GEMS extensions to optimize big graph analytics. GEMS is scoped to be a fast data analysis tool for big graphs instead of a general graph-mining tool. It is specifically designed to explore and exploit big-graph properties with significantly enhanced computation performance.

Table 1 highlights a selected set of foundational SPARQL commands that extract information such as graph properties and feature distributions [1]. We avoid SPARQL queries that are semantic-based, such as relying on type-inference (e.g., rdf:type and rdfs:subClassOf) [26], which allows us to adapt our techniques to non-RDF graphs. Using the retrieval results from Table 1, GEMS can further report additional results such as density (i.e., # of vertices divided by # of edges) of both ontology and instance vertices and edges.

Table 1: SPARQL queries with customized GEMS extensions.

	SPARQL	Return/Support
Properties	SELECT (COUNT (DISTINCT ?s) AS ?no) WHERE {{?s ?p ?o} UNION {?o ?p ?s}}	# of vertices
	SELECT (COUNT (?p) AS ?no) WHERE {?s ?p ?o}	# of edges
	SELECT (COUNT (DISTINCT ?s) AS ?count) WHERE {?s a ?class}	# of vertices with ≥ 1 type
Counts	SELECT (COUNT (DISTINCT ?s) AS ?no) WHERE {?s ?p ?o}	# of subject vertices
	SELECT (COUNT (DISTINCT ?o) AS ?no) WHERE {?s ?p ?o};	# of object vertices
	SELECT (COUNT (DISTINCT ?p) AS ?no) WHERE {?s ?p ?o}	# of predicates
Distributions	SELECT ?p (COUNT (?s) AS ?count1) WHERE {?s ?p ?o} GROUP BY ?p ORDER BY DESC (?count1)	Predicate distribution
	SELECT ?s (COUNT (?p) AS ?count1) {?s ?p ?o} GROUP BY ?s ORDER BY DESC (?count1)	Subject distribution
	SELECT ?o (COUNT (?p) AS ?count1) {?s ?p ?o} GROUP BY ?o ORDER by DESC (?count1);	Object distribution

5.3 High-Performance Computing System

Our system prototype is powered by an HPC system known as PUMA. Located at PNNL, PUMA is a Linux cluster comprising 32 nodes of Intel Xeon E5-2680 processors running at 2.8 GHz, eight Intel Xeon Phi 5110P coprocessors, and eight NVidia K40 GPUs. Each of the 32 nodes contains 20 physical cores, with a total of 640 cores. The system operates on both FDR InfiniBand and traditional Gigabit Ethernet networks.

PUMA is particularly equipped with abundant memory to meet the unique compute demand of in-memory applications such as big-graph analytics. It is known that "Graph problems that fit in memory can leverage excellent advances in architecture and libraries" [4]; PUMA has 24.5 TB of memory installed, divided evenly across the 32 nodes (i.e., 768 GB per node).

New hardware is currently being acquired to improve the performance of the PUMA infrastructure. We estimate that we will need ~24TB of memory to fit a trillion-edge graph (assuming 24 byte per graph edge or s-p-o triple) plus another ~50TB for indexing and intermediate computation space to operate. When the system is fully established, it is promised that exploring big graphs using GEMS on PUMA will scale to O(1T) graph edges [10][20].

6 System Prototype

This section gives a design overview of the proposed peek-and-filter paradigm and describes its implementation in detail.

6.1 System Design Overview

Figure 4 depicts an overview of the peek-and-filter implementation in the system prototype. On the left (red rectangle) is the graph database backend that runs GEMS on PUMA. On the right (orange rectangle) is a visual analytics frontend that runs T.Rex on a desktop computer. After a graph with tens of billions of edges (left) is ingested into a graph database, a user can peek-and-filter graph subsets using SPARQL commands in GEMS and send up to millions of graph edges (top) downstream to T.Rex (right) for visualization. Within the T.Rex frontend (i.e., inside the orange rectangle), there is another "peek and filter, then visualization" loop for users to fine-tune their explorations repeatedly until a final answer such as a vertex or a subgraph is identified (on the far right).

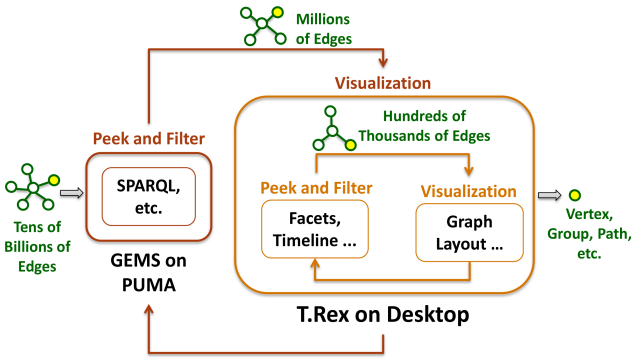

Figure 4: System design overview.

A major difference between the two peek-and-filter implementations in Figure 4 is that the one within T.Rex (orange rectangle) provides more powerful interactive visualization and analytical capabilities that can only be scaled up to hundreds and thousands of edges on a panel display. In other words, we use GEMS to address challenges 1 (size) and 4 (complexity) and T.Rex to address challenges 2 (cognition) and 3 (interaction) as described earlier in Section 3.

6.2 Graph Database Backend

GEMS [10][20][29] is a customized graph database system that has continued to evolve as algorithms are developed and hardware is introduced. As of this writing, GEMS is able to preprocess graphs with 25 billion edges in less than two hours using only 16 of the 32 nodes on PUMA. The preprocessing step 1) ingests the graph in n-triples format as an ASCII file and 2) creates both s-p-o and o-p-s indices that facilitate SPARQL queries afterward.

GEMS has a web-based interface that allows users to enter SPARQL commands and a second command-line interface that supports mainly software development and testing on PUMA.

6.3 Visual Analytics Frontend

T.Rex [27] is a visual analytics system for exploring transactional data that can be formulated conceptually as a graph, such as NetFlow data. Using a suite of visualization views along with a high-performance database server, T.Rex seamlessly supports the peek-and-filter visual analytics paradigm described in Section 4.2.

Figure 5: A screenshot of T.Rex analyzing a dataset with 6,939,698 NetFlow records or 627,572,995 triples.

Figure 5 shows a screenshot of T.Rex in action. Individual visualization views (A through K in the figure) can be repositioned, resized, or undocked from the main window. The Summary view (A) provides high-level contextual information, such as data size, current selection size, and, optionally, statistics (i.e., data peeking). This view also contains buttons to filter the dataset or reverse the selection (i.e., data filtering). Reversing the selection is a powerful tool to quickly select all the unselected records during exploration.

T.Rex's Facets view (B) is only applied to categorical data, such as an IP address. This view is designed to show frequently reoccurring values from dataset columns and what other values co-occur in the other columns of the dataset. The histograms embedded within the facets provide temporal awareness of these co-occurrences, as depicted in Figure 3.

The Graph view (C) uses a highly customized multiscale graph layout algorithm similar to the one implemented in GreenHornet [31] for big graph visualization. A user can select any two data columns to use as vertices and indicate if the drawing is a bi-partite graph. This view shows connectivity patterns, such as one-to-one, one-to-many, high connectedness, or islands of communication.

The Scatterplot view (D) plots a numeric field against either another numeric field or a date/time field. The Matrix view (E) shows highly correlated pairs of values contained in two data columns of the user's choosing.

The Group view (F) allows the grouping of a set of selected records and the inclusion of a group color that identifies the corresponding group members in the Graph and Scatterplot views. The group colors provide an effective way to specify different models for reasoning or organization and to perform analysis at a group level rather than at a record level.

The Timeline view (G) is a histogram visualization for temporal analysis and data selection. A corresponding Time Player view (H) animates the selected data in the Graph and Scatterplot views according to the user-defined step increment.

The Data Grid view (I) is a spreadsheet-like table browser. Records with a large number of columns can be inspected individually using the Record Viewer (J).

The Filters and Dataset Column views (K) allow selections of both table rows and columns using one or more filters. When a dataset is filtered, the views of the T.Rex application hide any records that have been removed by the filter. Filters can be created to focus on data subsets, excluding records that are not part of the active analytic context.

T.Rex was developed using a client/server architecture to maximize the user interaction and visual analytics capabilities and to use the processing power of a backend server, such as GEMS. T.Rex is a desktop-installed Java application developed using the Eclipse RCP framework. Each visualization is a plugin, giving added flexibility for custom deployments that require only a subset of the full functionality of the tool. Interested readers are encouraged to watch our T.Rex video [27] for additional examples.

7 Peek-and-Filter Demonstrations

We present a case study that demonstrates a series of peek-and-filter visual analytics tasks using a modest-sized NetFlow graph with ~6.9M records or ~627M triples provided by the IEEE VAST 2013 Challenge [15]. Numerous answers of the challenge have been posted online and published in the literature since the 2013 contest. We use the NetFlow dataset in our discussion here not to solve the challenge but to demonstrate how one can use the peek-and-filter approach in a semi-realistic cyber-analytics setting.

Figure 6 describes a series of "peek-and-filter then visualization" tasks to analyze the graph. Tasks highlighted in orange rectangles

are carried out by GEMS on PUMA and the ones in green are executed by T.Rex on a desktop computer.

The first step involves GEMS ingesting an RDF version of the VAST dataset. It takes GEMS 134 seconds to create the graph database from the dataset using 16 PUMA nodes. Using 8 PUMA nodes, GEMS ingests the dataset in 207 seconds. And with only 4 PUMA nodes, GEMS requires 364 seconds to ingest the dataset. Eight tasks (T1-T8) are illustrated in Figure 6 and explained below.

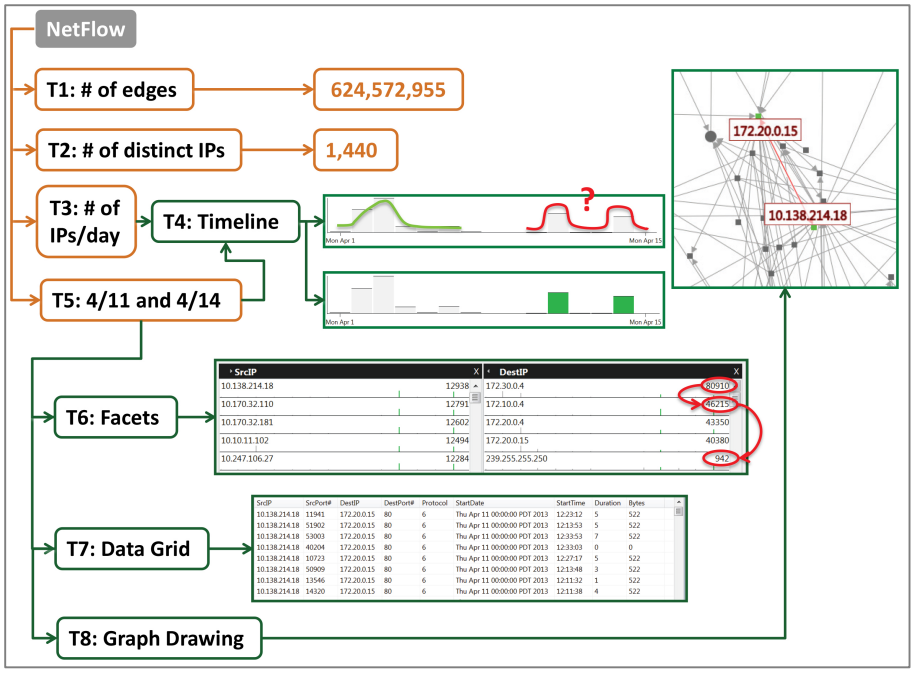

Figure 6: A graphical illustration of a peek-and-filter demonstration.

T1. After the graph database is created, the first knowledge learned about the graph is probably its size. The SPARQL command below reports that there are 624,572,955 edges in the graph database. Obviously, there are too many edges for an effective graph drawing visualization.

```
SELECT (COUNT (?p) AS ?no) {?s ?p ?o}
```

T2. For a NetFlow dataset with potentially up to millions of edges between a pair of IP addresses, perhaps a more interesting question to ask is how many distinct IP addresses are in the graph. The following SPARQL command reports that there are only 1440 unique IP addresses, implying that the graph has 1440 vertices.

```
SELECT (COUNT (DISTINCT ?s) AS ?no) {{?s ?p ?o}
UNION {?o ?p ?s}}
```

T3. The lower number of IP addresses in T2 suggests that we may be able to explore the graph using some basic visualization. The first visualization that comes to mind is a temporal distribution of the number of IP addresses per day, which could be computed using the following SPARQL command.

```
SELECT ?c (count(?c) as ?cnt) WHERE {{?a
<tag:pnnl.gov,2015:NF#srcIP> ?c} UNION {?a
<tag:pnnl.gov,2015:NF$dstIP> ?c}} GROUP BY ?c
```

T4. The results of T3—15 integers for 15 days—are sent to T.Rex's Timeline view for temporal visualization. The distribution quickly exposes two distinct structures in two consecutive weeks—a skewed pattern (green in T4) in the first week and a twin-peak pattern (red in T4) in the second week. The latter resembles a potential brute force "scan first and then attack" effort.

T5. To better understand the potential anomalies between the two peak days in the second week, all the actors (or IP addresses) involved are extracted from the graph database using the following SPARQL command for further investigation on T.Rex. The selected days are highlighted in green in the histogram of the Timeline view.

```
SELECT * WHERE {{?a <tag:pnnl.gov,2015:NF#stDate>
"2013-04-11"} UNION {?a
<tag:pnnl.gov,2015:NF#stDate> "2013-04-14"}}
```

T6. T.Rex's Facets view is used in this task to answer questions such as who are the major actors (either source or destination IPs) involved in the two-day period. Of the five most frequently visited destination IPs, the visualization show that the first IP (172.30.0.4) was connected twice as frequently as the second one (172.10.0.4) and about 90 times more than the fifth one (239.255.255.250), as highlighted by the red arrows in Figure 6.

T7. The numbers identified in T6 trigger a closer inspection of the original NetFlow records using T.Rex's Data Grid view, which is a spreadsheet with enhanced notational features. After scrolling down a few pages, we observe that records with the few IPs identified in the T6 vary little and a scrolling list is not the right tool to explore the hidden structures.

T8. Based on the findings in T7 and observations made earlier such as the graph size, we determine that it is now appropriate to use T.Rex's Graph view to interactively explore the layout of the extracted graph.

8 Computation Performance

We use the Berlin SPARQL Benchmark (BSBM) dataset generator and accompanying SPARQL queries [3] to demonstrate the computation performance of GEMS that powers the backend of our system prototype. BSBM provides elaborate datasets and advanced SPARQL queries that go beyond simple database retrieval and data lookup but instead ask questions such as "list the top 10 products most similar to a specific product, rated by the count of features they have in common."

Because the primary design goal of GEMS is to extract a reasonable amount of graph data for interactive visual analytics on T.Rex, we focus our computation performance study on *graph property* queries that facilitate the data extraction. Table 2 shows the computation results of a 25B-edge BSBM benchmark graph. It takes less than two hours for GEMS to ingest the data and set up the graph database for queries using only16 nodes on PUMA. We note that graph data at this scale cannot fit in to a single system's memory, so either paging or communication is required, which imposes a significant burden on overall computation time.

Table 2: Query times for different SPARQL queues running on PUMA using 16 nodes.

Query	Time (s)
# of vertices*	2109
# of edges*	341
# of subject vertices*	1034
# of object vertices*	974
# of predicates*	798
Predicate distribution*	1180
Highest out-degree: SELECT ?s (COUNT (?p) as ?c) {?s ?p ?o} GROUP BY ?s ORDER BY DESC (?c) LIMIT 100	1496

*The corresponding SPARQL commands are described in Table 1.

The time results in Table 2 are generated using only 50% (16 of the 32 nodes) of available CPUs on PUMA. Additionally, the current implementation of GEMS doesn't use the eight NVidia K40 GPUs that, with Tesla accelerators, can be up to 10 times faster than with CPUs (Intel Xeon E5-2680) alone when processing data. With the combination of our progress on supporting larger datasets and current hardware trends, we expect to support one trillion edges in the very near future.

9 Discussion

We discuss major contributions of the paper, lessons learned, and ongoing work that addresses the challenges ahead.

9.1 Major Contributions

The paper presents a non-traditional peek-and-filter paradigm for big graph visual analytics and a working system prototype that facilitates the exploration of a graph database of unprecedented size for visualization and analytics. Here we summarize our contributions from three different perspectives:

Big Data Visual Analytics – Conventional wisdom in visualization such as Shneiderman's visual information-seeking mantra will fall short when applied to big graph visual analytics problems. In our case, merely accessing an entire web-scale graph within a reasonable timeframe will be a stretch; getting a productive overview of the graph in visual form will be unthinkable because of the cognitive- and system-related challenges described in [30]. The peek-and-filter approach provides a viable solution to address some of these challenges found in big data visual analytics.

Big Data Computation Support – Our high-performance graph database system, GEMS, takes 1) less than two hours to ingest a graph database from a RDF graph with ~25 billion edges and 2) about 10-30 minutes to query against the graph database afterward. Compatible technologies reported in the literature would have required days to just ingest graphs of similar sizes [10].

Practicality and Applicability – Our desktop-based frontend, T.Rex, allows users to access big graph databases through GEMS and visually explore millions of transactional records (NetFlow in our case) extracted from the database. Because much of the heavy-lifting computational work is done on the HPC backend, we can run T.Rex on a modest computer such as a first-generation Microsoft Surface Pro tablet, as depicted in Figure 7.

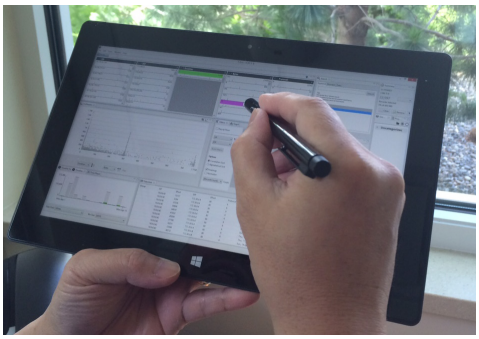

Figure 7: Running our big graph visual analytics prototype on a Microsoft Surface Pro Tablet.

9.2 Lessons Learned

Our lessons learned discussion focuses on three main areas: graph database, I/O bound, and visualization scalability.

Graph database – We use RDF to address many big-graph query issues brought by the costly schema expansion of a relational database. As the underlying graph size approaches web-scale ranges, we encounter thresholds beyond which it is necessary to distribute the graph across multiple disjoint memory systems as the only viable option for an in-memory database. Running graph search and retrieval computations on a distributed cluster adds performance pressure on the interconnection network and can slow the computation by several orders of magnitude.

I/O bound – Once in the realm of needing a distributed cluster platform, there are two major challenges to achieve good performance: latency of referencing data stored in a remote node and interconnection network throughput. The GEMS system overcomes the latency by employing a massive multithreading approach. Once a request to reference data in a remote node's memory is made, the task is moved aside and another task is brought into the core to work on. As long as there are enough tasks so that there is always something useful for a core to be doing, the latency is "tolerated." GEMS also uses well-known small message aggregation techniques to efficiently use the interconnection network in spite of the tendency for graph computation to involve small memory requests from remote nodes that result in a large number of small messages being sent out on the network.

Visualization scalability – To our surprise, the biggest and most immediate challenge facing T.Rex users and developers is not the very limited (and fixed) number of display pixels but the database response time that powers the query graphics from the backend. We believe that we can continue to increase the data volume and analysis capacity of T.Rex by first addressing the previous two lessons learned (i.e., graph database and I/O bound) and only then considering a new generation of visualization technology that will optimize the overall user experience.

9.3 Ongoing Work

To embrace a wider range of graph database applications, we are looking at using a *property graph model* [21] (attributed, directed, multi-graph) as the backend to our workflow. Such a change will bring new challenges to the T.Rex part of our strategy because the user will want the ability to see all of an edge's attributes and associated values. Displaying all these attributes and values will result in increasing the memory footprint of a subgraph sent from the backend to the T.Rex frontend.

The more general property graph data model will affect the graph search and retrieval steps by increasing the demand on the interconnection network. To offset the size of an edge that must transit the interconnection network, we are exploring how much of a reduction we will get in the number of vertices and edges by moving from an RDF graph to a property graph representation.

10 Conclusion

The paper presents a web-scale graph visual analytics paradigm and a proof-of-concept system prototype that, based on the computation and analytical results reported in this paper, will potentially be scaled to a trillion edges in the near future. We believe that the peek-and-filter approach is a viable option to address some of the web-scale graph exploration and analytics challenges. We will

continue to report the R&D progress and results of our web-scale graph visual analytics work at this venue in the future.

ACKNOWLEDGMENTS

This work has been supported in part by the U.S. Department of Defense (DoD) and other Federal agencies. The Pacific Northwest National Laboratory is managed for the U.S. Department of Energy by Battelle Memorial Institute under Contract DE-AC05-76RL01830.

REFERENCES

[1] Sinan al-Saffar, Cliff Joslyn, and Alan Chappell, "Graph-o-scope Concept," Technical Report PNNL-20137, Pacific Northwest National Laboratory, 2011.

[2] Apache Giraph, http://giraph.apache.org/.

[3] Berlin SPARQL Benchmark (BSBM), http://wifo5-03.informatik.uni-mannheim.de/bizer/berlinsparqlbenchmark/.

[4] Paul Burkhardt and Chris Waring, "An NSA Big Graph Experiment," *Technical Report NSA-RD-2013-056002v1*, U.S. National Security Agency, May 20, 2013. See also http://www.pdl.cmu.edu/SDI/2013/slides/big_graph_nsa_rd_2013_56002v1.pdf.

[5] Paul Burkhardt, "Big Graph," *The Next Wave*, 20(4), 2014. See also https://www.nsa.gov/research/tnw/tnw204/article3.shtml.

[6] Hank Childs, Kwan-Liu Ma, Hongfeng Yu, Brad Whitlock, Jeremy Meredith, Jean Favre, Scott Klasky, Norbert Podhorszki, Karsten Schwan, Matthew Wolf, Manish Parashar, and Fan Zhang, "In Situ Processing," *High Performance Visualization-Enabling Extreme-Scale Scientific Insight*, pages 171-198, Oct. 2012.

[7] Avery Ching, "Graph Analysis with One Trillion Edges on Apache Giraph," *Strata 2014*, Santa Clara, CA, Feb 2014.

[8] Avery Ching, "Scaling Apache Giraph to a trillion edges," Facebook, 2015. See also https://www.facebook.com/notes/facebook-engineering/scaling-apache-giraph-to`-a-trillion-edges/10151617006153920.

[9] Gephi, https://gephi.github.io/.

[10] Vito Giovanni Castellana, Alessandro Morari, Jesse Weaver, Antonino Tumeo, David Haglin, Oreste Ville, and John Feo, "In-Memory Graph Databases for Web-Scale Data," *IEEE Computer*, 48(3):24-35, IEEE CS Press, March 2015.

[11] Steffen Hadlak, Hans-Jörg Schulz, and Heidrun Schumann, "In Situ Exploration of Large Dynamic Networks," *IEEE Transactions on Visualization and Computer Graphics*, 17(12): 2334-2343, IEEE CS Press, December 2011.

[12] Steffen Hadlak, Heidrun Schumann, and Hans-Jörg Schulz, "A Survey of Multi-faceted Graph Visualization," in Ronald Lngner, Timo Luks, Anette Schlimm, Gregor Straubem and Dirk Thomaschke, editors, *Eurographics Conference on Visualization (EuroVis) 2015 State of the Art Reports*, The Eurographics Association, 2015.

[13] Minyang Han, Khuzaima Daudjee, Khaled Ammar, M. Tamer Ozsu, Xingfang Wang, and Tianqi Jin, "An Experimental Comparison of Pregel-like Graph Processing System," *Proceedings of the VLDB Endowment*, 7(12):1047-1058, 2014.

[14] David Harel and Yehuda Koren, "A Fast Multi-Scale Method for Drawing Large Graphs," *Proceedings of the 8th International Symposium on Graph Drawing GD 2000*, pages 183-196, Springer-Verlag, 2000.

[15] IEEE VAST Challenge 2013, http://vacommunity.org/VAST+Challenge+2013.

[16] Zuhair Khayyat, Karim Awara, Amani Alonazi, Hani Jamjoom, Dan Williams, and Panos Kalnis, "Mizan: A System for Dynamic Load Balancing in Large-scale Graph Processing," *Eurosys '13*, Prag, Czech Republic, April, 2013.

[17] Yucheng Low, Joseph Gonzalez, Aapo Kyrola, Danny Bickson, Carlos Guestrin, and Joseph M. Hellerstein, "GraphLab: A New Framework for Parallel Machine Learning," *CoRR abs/1006.4990*, 2010.

[18] Yucheng Low, Joseph Gonzalez, Aapo Kyrola, Danny Bickson, Carlos Guestrin, and Joseph M. Hellerstein, "GraphLab: A New Framework for Parallel Machine Learning," *CoRR abs/1408.2041*, 2014.

[19] Grzegorz Malewicz, Matthew H. Austern, Aart J.C. Bik, James C. Dehnert, Ilan Horn, Naty Leiser, and Grzegorz Czajkowski, "Pregel: A System for Large-Scale Graph Processing," *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data SIGMOD 2010*, pages 135-146, ACM, 2010.

[20] Alessandro Morari, Vito Giovanni Castellana, Oreste Villa, Antonino Tumeo, Jesse Weaver, David Haglin, John Feo, and Sutanay Choudhury, "Scaling Semantic Graph Databases in Size and Performance," *IEEE Micro*, 34(4):16-26, IEEE CS Press, July 2014.

[21] Property Graph Model, http://neo4j.com/developer/graph-database/#property-graph.

[22] Renoir, http://nsabackups.com/research/tech_transfer/fact_sheets/renoir.shtml.

[23] Resource Description Framework (RDF), http://www.w3.org/RDF/.

[24] Randall Rohrer, Celeste Lyn Paul, and Bohdan Nebesh, "Visual Analytics for Big Data," *The Next Wave*, 20(4), 2014. See also https://www.nsa.gov/research/tnw/tnw204/article4.shtml.

[25] Semih Salihoglu and Jennifer Widom, "GPS: A Graph Processing System," *Proceedings of the 25th International Conference on Scientific Statistical Database Management*, ACM Press, July 2013.

[26] SPARQL Query Language for RDF, http://www.w3.org/TR/rdf-sparql-query/.

[27] T.Rex, https://www.youtube.com/watch?v=GSPkAGREO2E.

[28] Tatiana Von Landesberger, Arjan Kuijper, Tobias Schreck, Jörn Kohlhammer, Jarke Van Wijk, Jean-Daniel Fekete, and Dieter Fellner, "Visual Analysis of Large Graphs: State-of-the-Art and Future Research Challenges," *Computer Graphics Forum*, 30(6):1719-1749, Wiley-Blackwell Publishing, 2011.

[29] Jesse Weaver, Vito Giovanni Castellana, Alessandro Morari, Antonino Tumeo, Sumit Purohit, Alan Chappell, David Haglin, Oreste Villa, Sutanay Choudhury, Karen Schuchardt, and John Feo, "Toward a data scalable solution for facilitating discovery of science resources," *Parallel Computing*, 40(10):682-696, Elsevier, 2014.

[30] Pak Chung Wong, Han-Wei Shen, Chaomei Chen, Chris Johnson, and Robert Ross, "Top Ten Challenges in Extreme-Scale Visual Analytics," *IEEE Computer Graphics and Applications*, 32(4):63-67, IEEE CS Press, July 2012.

[31] Pak Chung Wong, Patrick Mackey, Kristin A. Cook, Randall M. Rohrer, Harlan Foote, and Mark Whiting, "A Multi-Level Middle-Out Cross-Zooming Approach for Large Graph Analytics," *Proceedings IEEE Symposium on Visual Analytics Science and Technology (VAST) 2009*, pages 147-154, IEEE CS Press, October 2009.

[32] Kai Xu, Chris Rooney, Peter Passmore, Dong-Han Ham, and Phong H. Nguyen, "A User Study on Curved Edges in Graph Visualization," *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2449-2456, IEEE CS Press, December 2012.

[33] Xiaoru Yuan, Limei Che, Yifan Hu, and Xin Zhang, "Intelligent Graph Layout Using Many Users' Input," *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2699-2708, IEEE CS Press, December 2012.