iQCAR-demo: A demonstration of an Inter-Query Contention Analyzer for Cluster Computing Frameworks

Prajakta Kalmegh, Harrison Lundberg, Frederick Xu, Shivnath Babu, Sudeepa Roy
Duke University
{pkalmegh,hgl2,hx26,shivnath,sudeepa}@cs.duke.edu

ABSTRACT
Unpredictability in query runtimes can arise in a shared cluster as a result of resource contentions caused by inter-query interactions. iQCAR - inter Query Contention AnalyzeR is a system that formally models these interferences between concurrent queries and provides a framework to attribute blame for contentions. iQCAR leverages a multi-level directed acyclic graph called iQC-Graph to diagnose the aberrations in query schedules that lead to these resource contentions. The demonstration will enable users to perform a step-wise deep exploration of such resource contentions faced by a query at various stages of its execution. The interface will allow users to identify top-k victims and sources of contentions, diagnose high-contention nodes and resources in the cluster, and rank their impacts on the performance of a query. Users will also be able to navigate through a set of rules recommended by iQCAR to compare how application of each rule by the cluster scheduler resolves the contentions in subsequent executions.

ACM Reference Format:

1 INTRODUCTION
Large scale data analytics frameworks like Hadoop [10] and Spark [14] process a mix of short-running interactive BI (Business Intelligence) queries along with long-running ETL or batch analytics queries. Recurring queries co-exist with adhoc unplanned queries. Analytical SQL queries with varying resource utilizations over time share the cluster with machine learning, graph analytics, and data mining queries. In such shared clusters, resources are allocated to multiple tenants executing mixed workloads based on their priorities, SLAs (Service-Level Agreements), minimum share, etc. As such, resource allocations are controlled by the cluster scheduler using sophisticated arbitration techniques like capped capacities, reservations or use of scheduling policies like FAIR [13] or First-In-First-Out (FIFO). Most of these techniques rely on partitioning of available slots\(^1\) among the contending tenants. As a result, there are no guarantees on the usages of other resources like memory, disk IO, or network bandwidth for competing queries leading to inter-query resource interferences. This is a major concern in today’s clusters as performance issues due to resource contentions are often wrongly diagnosed, or are left unresolved due to lack of appropriate tools. It is, thus, important to analyze the victims (that we call a target query) and sources of these contentions (that we call source queries), especially identifying why and where a target query faces contentions from a source query. This can help a cluster administrator diagnose aberrations in resource allocations among tenants or devise alternative query placement strategies. For example, ranking the tenants based on their contention impact towards a target query can prove particularly useful to revisit the resource shares of these tenants.

\(^1\)We refer to a slot as the smallest unit of resource allocation. For example, its a CPU core in Spark and a combination of CPU and Memory in Hadoop.

Figure 1: iQC-Graph(V,E): The sub-graph in Explanations Layer is unique for each stage in Level-1

In this demonstration, we will present iQCAR - inter Query Contention AnalyzeR, a system to explore contentions faced by queries due to inter-query interactions on a cluster computing framework. iQCAR interface allows users to interact with a multi-level directed acyclic graph (DAG) to progressively unravel three levels of explanations; namely (i) Immediate Explanations (IE): identify disproportionate waiting times spent by a query blocked for a particular resource, (ii) Deep Explanations (DE): inspect this waiting time for every resource used by the query on all hosts where it executed, and finally (iii) Blame Explanations (BE): quantify the impact from concurrent queries towards the slowdown for a target query. Figure 1 presents the different levels of explanations in iQC-Graph. Additionally, users will be able to filter and navigate...
through cluster-level summary visual aids on: (a) high contention resources and nodes, (b) high impact causing source queries, and (c) high impact receiving target queries. Finally, users will be able to browse through a list of alternate schedule rules recommended by 1QCAR and compare the results of applying them in recurring execution of the workloads.

2.1 Multi-layered iQC-Graph
Level-0 of iQC-Graph consists of the target queries to be analyzed and Level-1 contains the stages of each target query. Level-5 and Level-6 constitute the concurrently running source stages and source queries respectively. The explanations layer consisting of Levels 2, 3 and 4 provides explanations of different forms and granularity. It enables us to connect the two ends of iQC-Graph with appropriate assessment of contention impact among all intermediate nodes and edges.

2.1.1 Immediate Explanations (IE): Level-2 vertices provide an explanation of the form 'how much time was spent by a stage waiting for a particular resource per unit of data processed'. For every stage $St_q$ at Level-1 of target query $Q_t$, we add an IE vertex at Level-2 for every resource (scheduling queue, CPU, Memory, Network, IO) used by stage $St_q$ and store the value of its wait-time for this resource per unit data processed as its Vertex Contribution (VC).

2.1.2 Level-3: Deep Explanations (DE): Level-3 captures the hosts responsible towards the corresponding disproportionality in the wait time components for every resource. That is, DEs keep track of the wait-time distributions per unit of data processed by stage $St_q$ for a specific resource $r$ on each host $h$ of execution. That is, for each 1E node in Level 2, we add $h$ DE nodes in Level 3 corresponding to all hosts on which the tasks of $St_q$ executed.

2.1.3 Level-4: Blame Explanations (BE): Blame Explanations is a novel contribution of 1QCAR. For each vertex $v$ in Level-3 (DE) corresponding to $St_q$, host $h$, and type of resource request $r$, if tasks of $St_q$ were concurrent with tasks of $P$ stages of other queries on host $h$, we add $P$ nodes $u$ in Level 4 and connect them to $v$. To compute the blame to be assigned to a concurrent stage, we first compute the blame from a source task $st$ (task of concurrent stage) to a target task $tt$ executing concurrently on host $h$ while using resource $r$ and use it to compute the VC of each BE node.

2.2 Architecture
1QCAR provides a cluster interface to analyze existing Spark workload execution logs, submit new Spark applications, or simulate an existing workload using TPCDS [3] benchmark queries.

wSEL and wSUB: The workload selection module (wSEL) allows users to select a pre-executed workload for analysis. To let users simulate a workload on Spark, the workload submission interface (wSUB) allows them to select a list of benchmark TPCDS queries, the order of these queries and finally their arrival schedule (fixed-delay or poisson or user-input start times). The users can also specify the interval (in seconds) for collecting the task execution metrics. By default, we collect metrics after the completion of each task, stage, job or query in Spark.

![Figure 2: 1QCAR Architecture](image-url)
the level to which the vertex belongs to, and are computed during the graph construction process as described in Section 2.1. We then perform a top-down pass on this graph to update the edge weights using the IF-calc, and next do a bottom-up pass to update the responsibility measure of each vertex using the DOR-calc. The Impact Factor (IF) of an edge gives the impact a source vertex of the edge has towards the end vertex. Degree of Responsibility (DOR) of a vertex is defined as the cumulative impact this vertex has towards a target query.

r1GEN and r1APP: The rule generator (r1GEN) module uses the consolidated contention measures from blameAnalyzer and outputs two types of rules: (i) Alternate Query Placement: r1GEN outputs the top-k aggressive queries (high-impact towards all selected target queries) and generates k rules that recommend placing each of these queries in a new pool with new recommended shares for each of them. (ii) QueryPriority Readjustment: r1GEN produces k priority rectification rules for each of the top-k affected target queries that suffered the highest impact, and top-k impacting source queries. The r1APP module is an extension to the Spark standalone scheduler that parses the rules and applies any active and applicable rules during a recurring execution of the workload.

iQCAR VIZ : The iQCAR visualization module is a web based front-end that enables users to (i) select a workload for analysis using our wlSEL module and visualize its key characteristics using the wlVIZ interface, (ii) explore summary of query execution and cluster performance using the perfVIZ interface that provides tips on selecting a single target query or a set of target queries for further exploration, (iii) delve into the task-level execution and wait-time distribution details for each query using the dmVIZ interface, (iv) perform a systematic deep exploration of the Neo4j iQC-Graph that lets users unfold explanations at various levels and analyze the relative impacts from concurrent queries using the grVIZ visualization aids, and (v) finally, use the r1VIZ module that lets users compare the results of applying a selected set of rules on the recurring execution of the workload. The iQCARViz interface also allows users to compare the impact of different monitoring intervals of data collection on our contention analysis metrics. Each of the visualization module uses the iQCARViz dataframes API to render plots dynamically based on user-input online using Plotly [8] tool.

3 DEMONSTRATION

The purpose of this demonstration is to (i) showcase the users the tedious process of contention analysis in the absence of iQCAR, (ii) enable users with a hands-on experience of using iQCAR for insightful analysis, and (iii) present users an opportunity to compare and contrast the results of applying the iQCAR rules on a benchmark workload’s performance. To achieve these goals, we will divide the demo in three segments, namely (a) manual analysis of pre-executed TPCDS benchmark execution, (b) deep exploration of contentions, and (c) analyze the output of iQCAR.

Figure 3: iQCAR visual aids for answering cluster-level contention analysis questions.

Figure 4: (a) Screen to select a workload for contention analysis. (b) Screen to perform a time-series analysis of iQC-Graph.

Setup: Users will be given two options to analyze a workload: (i) select a pre-executed microbenchmark workload, or (ii) submit a workload by selecting a set of TPCDS-queries along with their arrival pattern. A sample screen for this workload selection is shown in Figure 4a. The workloads will be executed on a 10-node cluster setup with Apache Spark 2.2 [14] and Hadoop 2.7 [10].

3.1 Segment 1: Manual Analysis

Users will be asked to answer one randomly selected multi-choice question on the contention faced by a query using the existing monitoring tools like Spark UI [9] and Ganglia [5]. This activity will demonstrate the tedious process of performing a manual contention analysis even on a small-size cluster.

3.2 Segment 2: Explore iQCAR

The next step in our demo is to explore the interface of iQCAR that will enable users answer questions like below divided broadly into the following three categories based on the level of contention details they provide:

Summary Questions: Users can choose to browse through a series of cluster summary visual aids for our sample workloads as illustrated in Figure 3.

- Q1: Which hosts in the cluster had the highest CPU contention?
- Q2: On which resource were all queries bottlenecked the most?
- Q3: Which queries are the victim of highest contention?
workload. Next, Figure 5b shows how users can choose whether to provide explanations for highest impact between any two queries, including features that let the users (i) highlight the path from a single source query to a target query with highest path weight (useful for providing explanations for highest impact between any two queries), (ii) display all paths that add up more than a certain threshold of time waiting, (iii) choose which queries are the cause of highest contention, (iv) choose combinations of a host, resource, and source queries to analyze impact on a single selected target query.

Target Query Performance Analysis Questions: Figure 5a shows how iQC-Graph provides a visual aid for users to select a target query for further deep exploration from a set of completed queries in a workload. Next, Figure 5b shows how users can choose whether they want to analyze contentions on (i) a single stage, (ii) all stages in the critical path (execution time dominating path), or (iii) all stages of a query using iQC-Graph. For performing a single-query analysis, users can drill-down to various levels of details by filtering on hosts, resources or specific stages of target and source queries as shown in Figure 5c to explore the impact on a particular target query $Q_t$. This interface allows users to answer questions like:

- **Q4:** Which queries are the cause of highest contention?
- **Q5:** On which user-selected combination of host $h$ and resource $r$, has a target query $Q_t$ (or its selected stages) spent maximum time waiting?
- **Q6:** Which queries are responsible for causing highest contention for a target query $Q_t$ (or its selected stages) on a user-selected combination of host $h$ and resource $r$?

Source Query Performance Analysis Questions: Finally, users can also perform a top-down analysis on iQC-Graph to draw insights on how a query impacts or causes contentions to others using various filters on its stages, hosts, and resource types. Due to space constraints, the screenshots for these visualizations are not shown.

- **Q7:** Which queries were affected most by the contention caused by a source query $Q_s$ (or its selected stages) on a user-selected combination of host $h$ and resource $r$?

The above visual aids help users to get answers to questions Q1 to Q7 rapidly. For users who want to diagnose each contention in detail, the iQC-Graph visualization powered by D3.js [4] provides a step-wise exploration opportunity. For instance, users can click on each node to progressively expand further levels of sub-graph. Users can also click on each vertex and edge to view the values of our contention analysis metrics (VC, IF, and OR). Other interface features will let the users (i) highlight the path from a single source query to a target query with highest path weight (useful for providing explanations for highest impact between any two queries), (ii) display all paths that add up more than a certain threshold of user-input impact value (useful to discover contention conditions beyond an acceptable threshold), and (iii) load source queries that impact the selected target queries only within an user-input time frame. Figure 4b shows the iQC-Graph for our example workload for the last scenario where user inputs the start and end times to input a time frame for impact analysis.

### 3.3 Segment 3: Analyze iQCAR results

The final segment in the demo will enable users to examine a set of rules output by the r1GEN module of iQCAR. Users will be able to compare the performance of the workloads (new runtime of each query, wait-times on all resources and/or hosts,) after application of the top-3 rules of each type. Users can also use the recommended priority to choose and apply the rules for a more real-time experience.

**Related Work:** The field of explanations has been studied in many contexts like analyzing job performance [11]. In [12], the authors present a general framework to analyze data-analytical workloads using blocked-time metric. We use this pedestrian to present iQCAR as a first systematic attempt towards exploration of different levels of explanations for resource contentions on cluster frameworks. See [6] for a detailed discussion on related work.

### REFERENCES


---

**Figure 5:** (a) Screen to aid users identify queries for further exploration. It shows the runtime hit of all queries compared to their unconstrained (no contention) execution time. (b) Screen to aid users identify stages of a selected query for further exploration. (c) iQC-Graph visualization to select combinations of a host, resource and source queries to analyze impact on a single selected target query.