concurrency

Description

Concurrency measures the number of events which have spans that overlap with the start of each event. Alternatively, this measurement represents the total number of events in progress at the time that each particular event started, including the event itself. This command does not measure the total number of events that a particular event overlapped with during its total span.

Syntax

concurrency duration=<field> [start=<field>] [output=<field>]

**Required arguments**

duration

**Syntax:** duration=<field>

**Description:** A field that represents a span of time. This field must be a numeric with the same units as the `start` field. For example, the duration field generated by the transaction command is in seconds (see Example 1), which can be used with the default of `_time` which is also in units of seconds.

**Optional arguments**

start

**Syntax:** start=<field>

**Description:** A field that represents the start time.

**Default:** `_time`

output

**Syntax:** output=<field>

**Description:** A field to write the resulting number of concurrent events.

**Default:** "concurrency"

Usage

An event X is concurrent with event Y if X.start is between Y.start and (Y.start + Y.duration)
If your events have a time that represents event completion and a span that represents the time before the completion, you need to subtract duration from the start time before the concurrency command:

```bash
... | eval new_start = start - duration | concurrency start=new_start
duration=duration
```

**Limits**

There is a limitation on quantity of overlapping items. If the maximum tracked concurrency exceeds max_count, from the [concurrency] stanza in limits.conf, a warning will be produced in the UI / search output, and the values will be clamped, making them potentially inaccurate. This limit defaults to 10000000 or ten million.

**Basic examples**

1. **Determine the number of overlapping HTTP requests**

Determine the number of overlapping HTTP requests outstanding from browsers accessing splunkd at the time that each http request begins.

This relies on the fact that the timestamp of the logged message is the time that the request came in, and the 'spent' field is the number of milliseconds spent handling the request. As always, you must be an 'admin' user, or have altered your roles scheme in order to access the _internal index.

```bash
index=_internal sourcetype=splunkd_ui_access | eval spent_in_seconds = spent / 1000 | concurrency duration=spent_in_seconds
```

2. **Calculate the number of concurrent events**

Calculate the number of concurrent events for each event and emit as field 'foo':

```bash
... | concurrency duration=total_time output=foo
```

3. **Use existing fields to specify the start time and duration**

Calculate the number of concurrent events using the 'et' field as the start time and 'length' as the duration:

```bash
... | concurrency duration=length start=et
```
Extended examples

1. Count the transactions that occurred at the same time

This example uses the sample data from the Search Tutorial. To try this example on your own Splunk instance, you must download the sample data and follow the instructions to get the tutorial data into Splunk. Use the time range All time when you run the search.

Use the duration or span of a transaction to count the number of other transactions that occurred at the same time.

```splunk
sourcetype=access_* | transaction JSESSIONID clientip startswith="view" endswith="purchase" | concurrency duration=duration | eval duration=tostring(duration,"duration")
```

- This search groups events into transactions if they have the same values of JSESSIONID and clientip. An event is the beginning of the transaction if the event contains the string "view". An event is the last event of the transaction if the event contains the string "purchase".
- The transaction command returns a field called duration.
- The transactions are then piped into the concurrency command, which counts the number of events that occurred at the same time based on the timestamp and duration of the transaction.
- The search also uses the eval command and the tostring() function to reformat the values of the duration field to a more readable format, HH:MM:SS.
To see the values in each transaction for the JSESSIONID, clientip, concurrency, and duration fields:

1. In the list of Interesting Fields, click the field name.
2. In the information box, for Selected, click Yes.
3. Select the next field in the list of Interesting Fields. The information box automatically refreshes. For Selected, click Yes.
4. Repeat these steps for every field you want to appear in the result list. The results should appear similar to the following image:

![Image of transaction list]

2. Count the purchases that occurred at the same time

This example uses the sample data from the Search Tutorial. To try this example on your own Splunk instance, you must download the sample data and follow the instructions to get the tutorial data into Splunk. Use the time range All time when you run the search.

Use the time between each purchase to count the number of different purchases that occurred at the same time.

```
sourcetype=access_* action=purchase | delta _time AS timeDelta p=1 | eval timeDelta=abs(timeDelta) | concurrency duration=timeDelta
```

- This search uses the `delta` command and the `_time` field to calculate the time between one purchase event (action=purchase) and the purchase event immediately preceding it.
- The search renames this change in time as `timeDelta`.
- Some of the values of `timeDelta` are negative. Because the `concurrency` command does not work with negative values, the `eval` command is used to redefine `timeDelta` as its absolute value (`abs(timeDelta)`).
- The `timeDelta` is then used as the `duration` for calculating concurrent events.
3. Calculate the transactions using the time between consecutive transactions

This example uses the sample data from the Search Tutorial. To try this example on your own Splunk instance, you must download the sample data and follow the instructions to get the tutorial data into Splunk. Use the time range **All time** when you run the search.

Use the time between each consecutive transaction to calculate the number of transactions that occurred at the same time.

```
sourcetype=access_* | transaction JSESSIONID clientip startswith="view" endswith="purchase" | delta _time AS timeDelta p=1 | eval timeDelta=abs(timeDelta) | concurrency duration=timeDelta | eval timeDelta=toString(timeDelta,"duration")
```

- This search groups events into transactions if they have the same values of **JSESSIONID** and **clientip**. An event is the beginning of the transaction if the event contains the string "view". An event is the last event of the transaction if the event contains the string "purchase".
- The **transaction** command returns a field called **duration**.
- The transactions are then piped into the **delta** command, which uses the `_time` field to calculate the time between one transaction and the transaction immediately preceding it.
- The search renames this change in time as **timeDelta**.
- Some of the values of **timeDelta** are negative. Because the **concurrency** command does not work with negative values, the **eval** command is used to redefine **timeDelta** as its absolute value (abs(timeDelta)).
- This **timeDelta** is then used as the **duration** for calculating concurrent transactions.
See also
timechart

Answers

Have questions? Visit Splunk Answers and see what questions and answers the Splunk community has using the concurrency command.