Simply Top Talkers

Jeroen Massar, Andreas Kind and Marc Ph. Stoecklin

© 2009 IBM Corporation

Motivation and Outline

- Need to understand and correctly handle dominant aspects within the overall flow of traffic
  - Top-k Problem
  - Optimize peering relationships (top autonomous systems)
  - Analyze congestion cases (top hosts)
- Technical challenge
  - Compute sorted top-n views from very large numbers of flow information records
  - No possibility to store individual counters per aspect components
- This presentation
  - Propose and analyze simple techniques to compute top-k listings for single and composed traffic aspects
  - Show that the techniques are suitable for in-memory aggregation databases

© 2010 IBM Corporation

Related Work

- E. D. Demaine, A. Lopez-Ortiz, and J. I. Munro. Frequency estimation of internet packet streams with limited space. In *Proceedings of the 10th Annual European Symposium on Algorithms*, pages 348–360, 2002.

- C. Estan and G. Varghese. New directions in traffic measurement and accounting: Focusing on the elephants, ignoring the mice. *ACM Trans. Comput. Syst.*, 21(3):270–313, 2003.

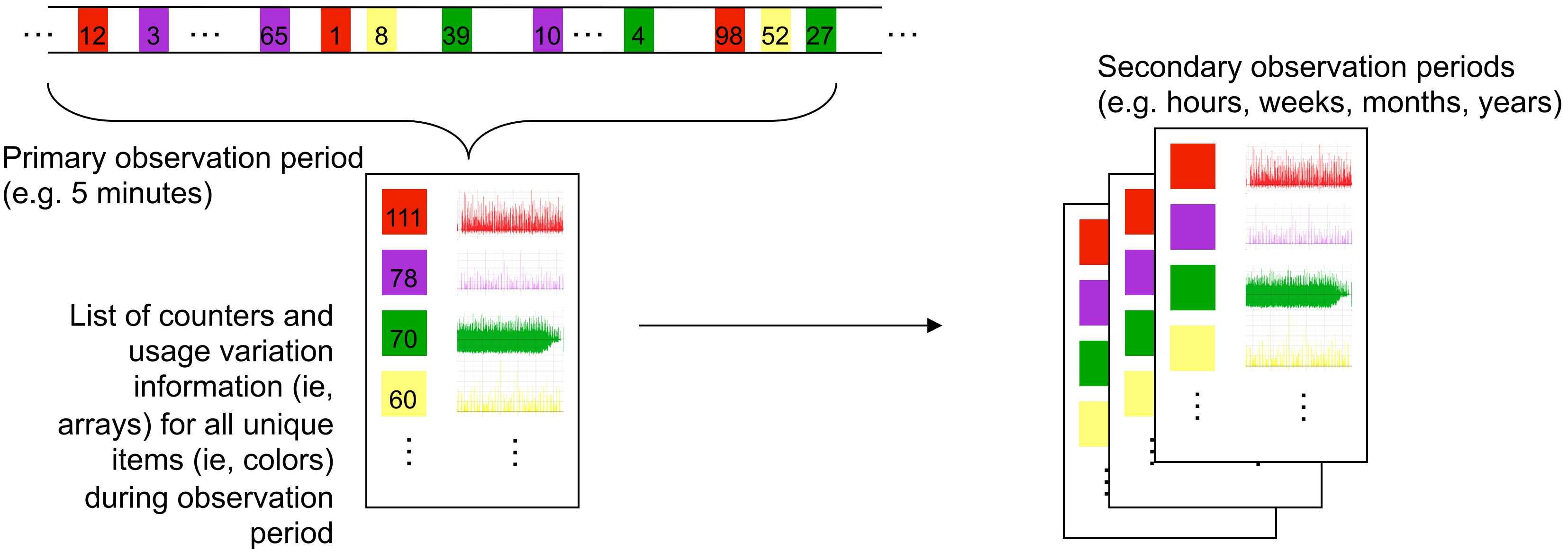
- Metwally A, Agrawal D, El Abbadi A (2005) Efficient computation of frequent and top-k elements in data streams. In: Proceeding of the 2005 international conference on database theory (ICDT' 05), Edinburgh, UK, pp 398–412

- X. Dimitropoulos, P. Hurley, and A. Kind: Probabilistic Lossy Counting: An Efficient Algorithm for Finding Heavy Hitters," ACM SIGCOMM Computer Communication Review, Jan. 2008.

- X. Dimitropoulos, M. Stoecklin, P. Hurley, and A. Kind: The Eternal Sunshine of the Sketch Data Structure," Elsevier Computer Networks, 2008.

© 2010 IBM Corporation

Top-k Problem



Primary observation period (e.g. 5 minutes)

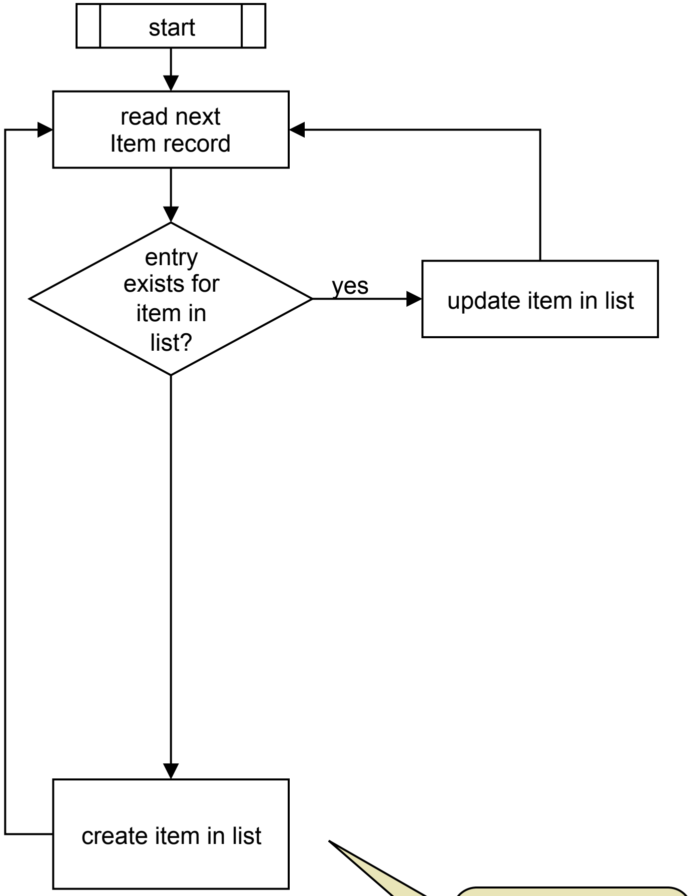
List of counters and usage variation information (ie, arrays) for all unique items (ie, colors) during observation period

Secondary observation periods (e.g. hours, weeks, months, years)

- Large number of unique items per observation period
  - – Large alphabet
  - – Counters and usage arrays don’t fit into memory

- How to cut back sorted list?
  - – Correct ordering
  - – Correct volume information for each item
  - – Reduce the cost of inserting new items (that anyway would not make it into the sorted top-k list)

One Threshold Scheme (*max1*)



start

read next
Item record

entry
exists for
item in
list?

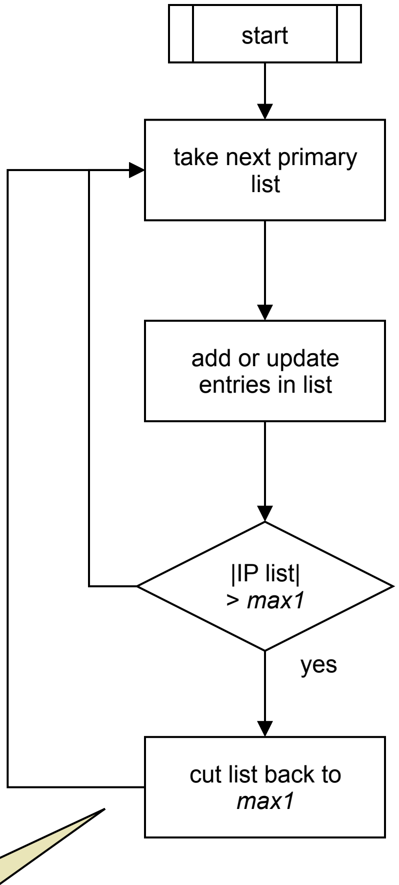
yes

update item in list

create item in list

Primary list can grows up to unique number of items

Merging
primary list into
secondary list



start

take next primary
list

add or update
entries in list

|IP list|
> *max1*

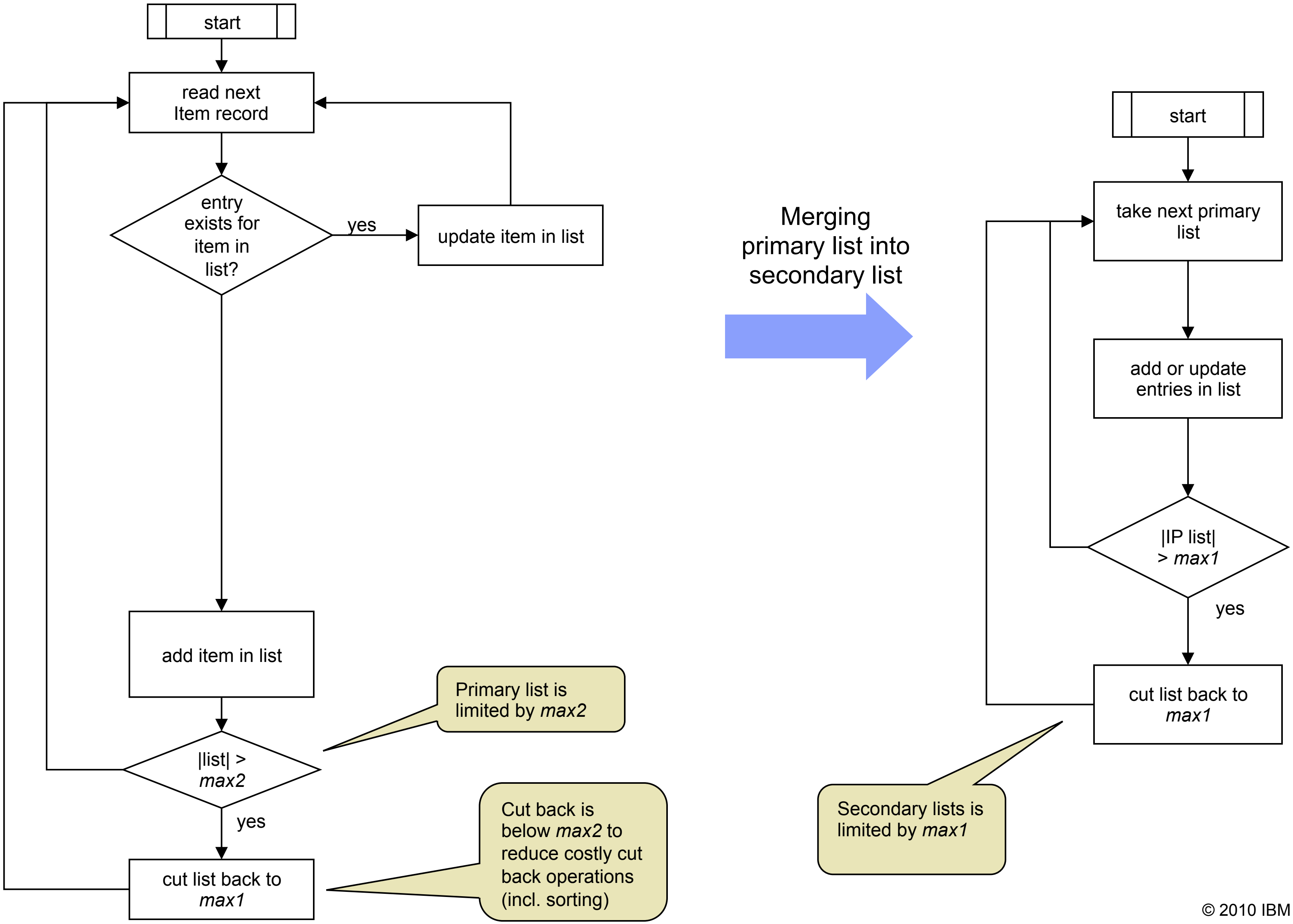
yes

cut list back to
max1

Secondary lists is limited by *max1*

© 2010 IBM Corporation

Two Threshold Scheme (*max1*, *max2*)



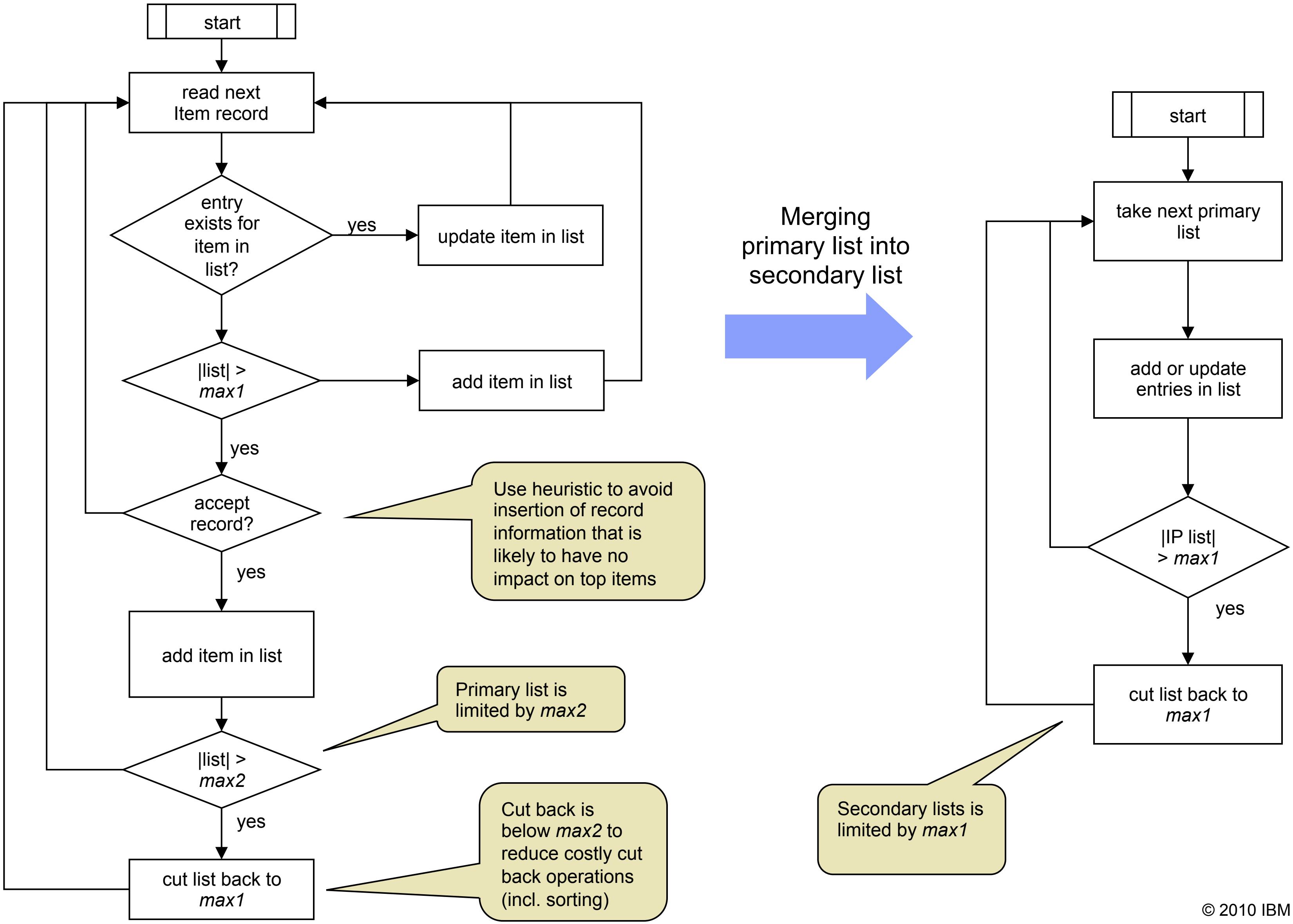
Merging primary list into secondary list

Primary list is limited by *max2*

Cut back is below *max2* to reduce costly cut back operations (incl. sorting)

Secondary lists is limited by *max1*

Two Threshold Scheme (*max1*, *max2*) with Heuristic



start

read next Item record

entry exists for item in list?

yes

update item in list

|list| > *max1*

add item in list

yes

accept record?

yes

add item in list

|list| > *max2*

yes

cut list back to *max1*

Use heuristic to avoid insertion of record information that is likely to have no impact on top items

Primary list is limited by *max2*

Cut back is below *max2* to reduce costly cut back operations (incl. sorting)

Merging primary list into secondary list

start

take next primary list

add or update entries in list

|IP list| > *max1*

yes

cut list back to *max1*

Secondary lists is limited by *max1*

Heuristics

- 0. Exact

- max1,max2 style:
- No Heuristic
- Short duration (< 1 second)
- Low Packet count in one flow (< 4 pkts)
- Bytes < total_bytes_seen/total_flows_seen (averaging)

- These heuristics are very simple and more importantly ‘cheap’ to implement memory and cpu wise.

© 2010 IBM Corporation

Data Set

- We picked three days of data collected at one of the IBM datacenters

- Did an expensive query: calculate the exact top K entries
- Fortunately there are machines with large amounts of memory and number of cores…

- Then we applied the defined heuristics to the data set, which run in near real-time as we don’t need a lot of memory or cpu power to store all the entries

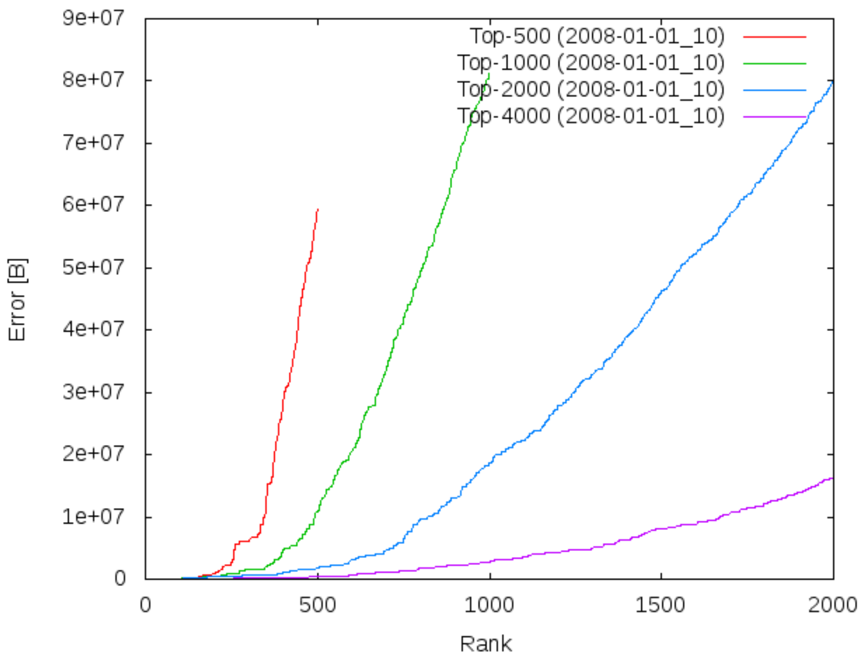
- One of the biggest advantages of course is that one can almost stay in-cache when one has proper cpu’s (4 or 8 MiB cache per core)

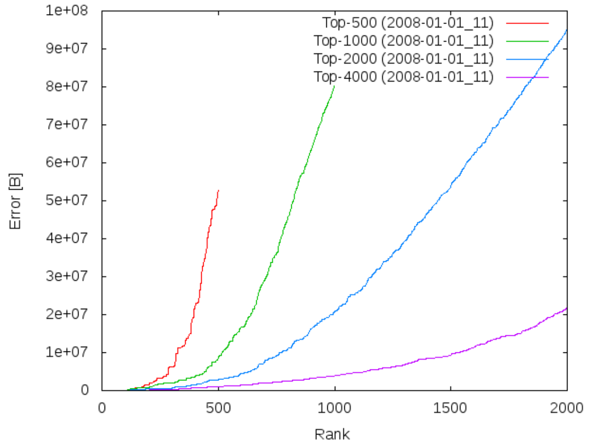
- results on next slides….

- Note that the dataset represents several Petabytes of network traffic, the ‘error’ rate in comparison is thus effectively very low…..

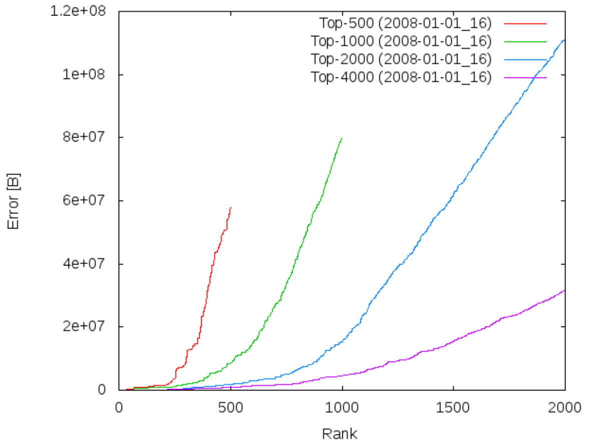
© 2010 IBM Corporation

Error rate for Top K without heuristic



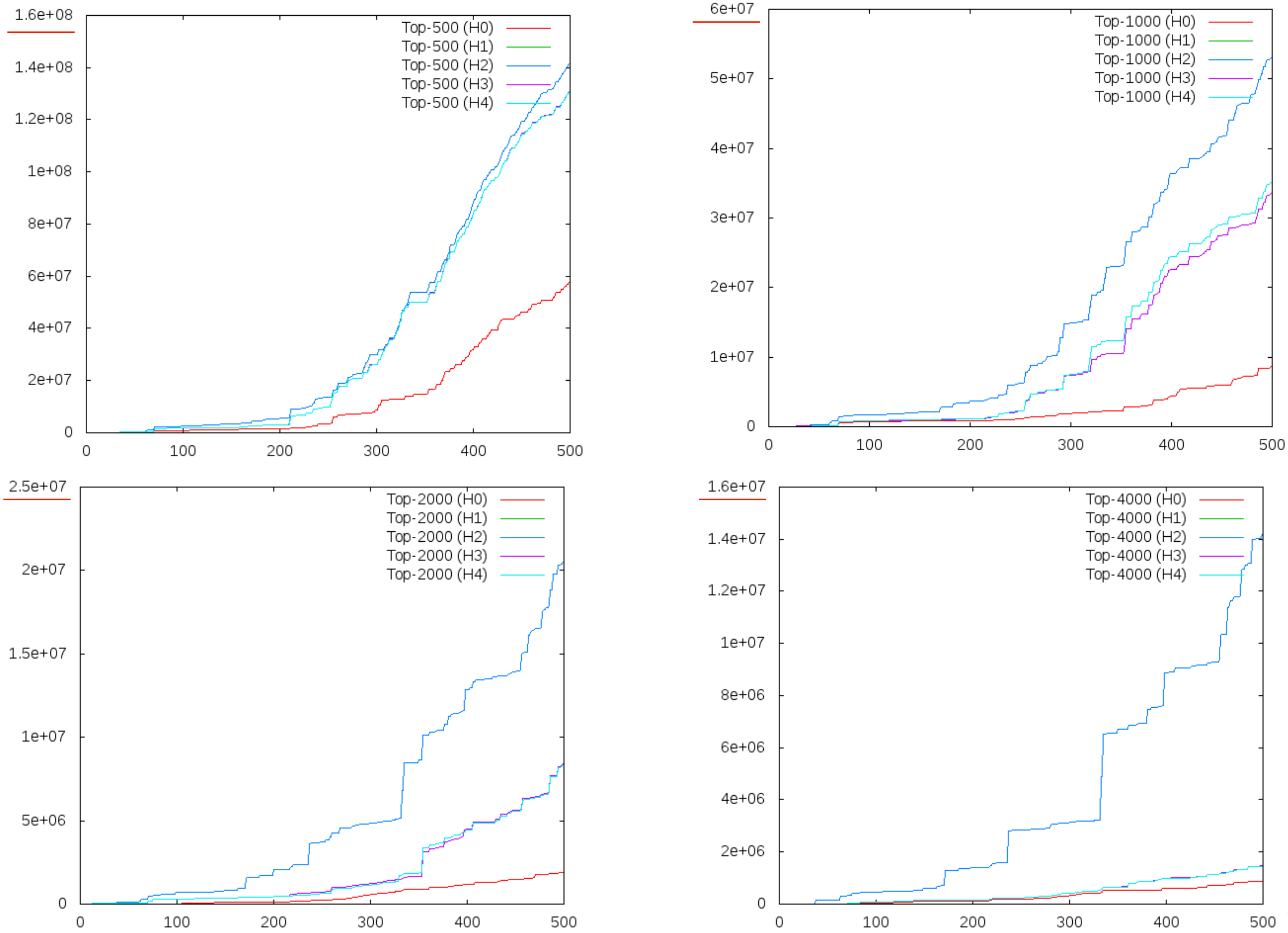


Amount of bytes that are not seen for top X items



© 2010 IBM Corporation

Error rate for Top K when applying various heuristics



Conclusion

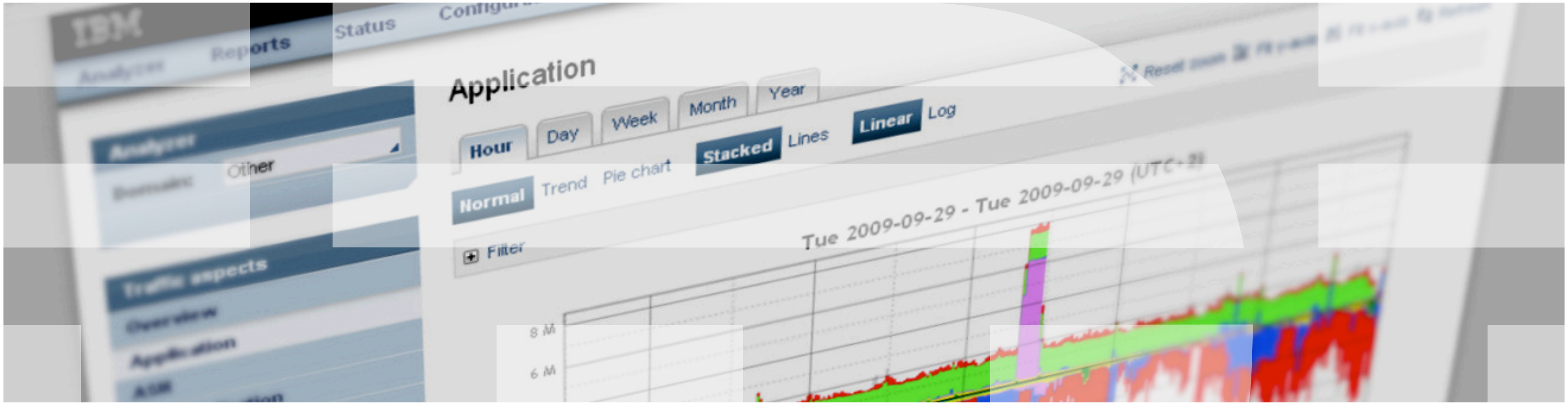
- Simple heuristics and thus cheap processing:
  - Differences are minimal for this ‘datacenter’ dataset, other datasets, eg for an enterprise network seem to prefer the ‘bytes < avg’ heuristic a bit more, which makes sense as it is hard to get in the top K unless you can at least beat the average.

- Two thresholds
  - max2 = 2 * max1
  - Correct order up to k = X * max1

© 2010 IBM Corporation

AURORA

http://www.zurich.ibm.com/aurora/



© 2010 IBM Corporation