



Beyond Streaming: Practical Models for Processing Large Data Sets

Sridhar Rajagopalan
IBM Almaden Research Center

Collaborators:
Matthias Ruhl, Mayur Datar, Gagan Agarwal, Eugene
Shekita, Kevin Beyer, Andrew Tomkins, Dan Gruhl,
Members of the ProjectWF Team

The problem

- Build a system and programming model for processing large datasets.
- Take away the complexities of dealing with large data.
 - Exceptions will occur.
 - Components (both hard and soft) will fail.
 - Data structures will exceed available memory.
- Approximate or incomplete results are usually good enough.

Commodity Computing Platforms

1. Disk based storage is plentiful and cheap.
 - Current price less than \$1.5 dollars/gigabyte.
2. Memory hierarchies are complicated and pipelines are deep.
 - Large gap between random and sequential access, even within main memory.
 - Random access in main is slower than sequential access to disk.
 - CPU is underutilized, especially in data intensive applications.
 - Where have all the cycles gone?
3. Parallelism is great in theory, but hard in practice.
 - Naïve data partitioning is the only practical option to keep system complexity under control.

The Streaming Model [Munro-Paterson]

- Data is seen as a sequence of elements.
 - Proposed as a model for large data sets, as well as data from sensors.
 - Issues:
 - Memory usage.
 - Passes required over the data.
 - Many variations.
-
- Sorting is hard. As are almost all interesting combinatorial/graph theoretic problems.
 - Exact computation of statistical functions are hard. Approximation is possible.
 - Relationships to communication complexity and information complexity.

External Memory [Aggarwal-Vitter]

- Two costs (multiple costs) of access to main and disk storage.
 - Page based to simulate the benefits of sequential access.
 - More realistic than the streaming model as a representation of commodity hardware.
-
- Not a good programming model. Needs the programmer to understand system specific parameters.
 - Does not account for the efficiency of the sequential access pipeline.

Sorting

- Sorting large a large data set on commodity hardware is a solved problem.
 - Google sorts more than 100 B terms in its index.
 - SPSort.
 - Penny Sort, Minute Sort.
- But Sorting well requires a great deal of care and customization to the hardware platform.
- What is the cost of indexing the Web? 2B documents, each with 500 words = 1 Trillion records. Cost of index build per Penny Sort is under 50 bucks.
- Networking speeds make sharing large quantities of streaming data possible.

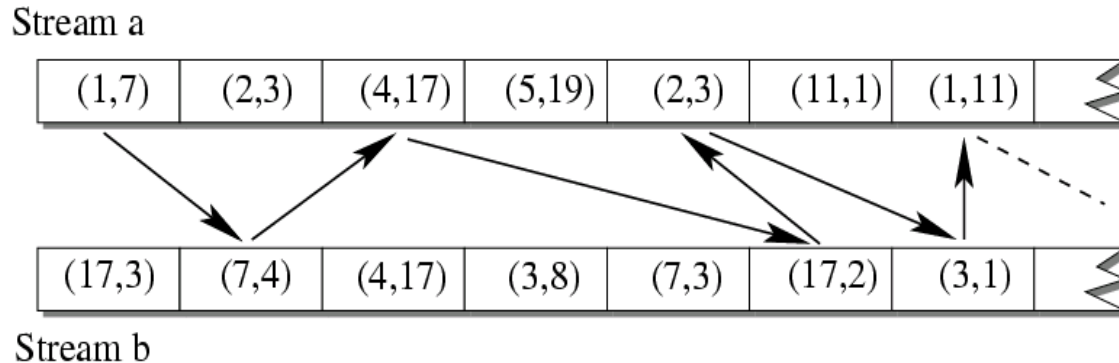
Model 1: Stream + Sort

- Basic multi-pass data stream model with access to a Sort box.
- Quite powerful.
 - Can do entire index build (including PageRank like calculations).
 - Spanning Tree, Connected Components, MinCut, STCONN, Bipartiteness.
 - Exact computations of order statistics and frequency moments.
 - Suffix Tree/Suffix Array build.
 - Red/Blue segment intersection.
 - So strictly stronger than just streaming.

Theorem : $NC \subseteq \text{StrSort}$

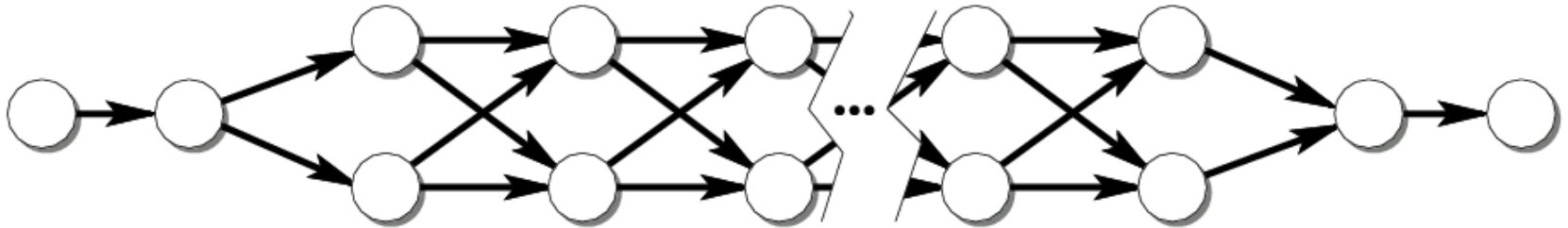
How powerful is StrSort?

- Not as powerful as P.
- Number of Sorting Passes is critical.
- Alternating sequence problem.



Model 2: Multiple Disks - Streaming Networks

- Computation is expressed as a network. Each node represents a streaming computation.



Relationships

- StrSort vs. Streaming Networks.
 - Issue is the number of concurrent streams available for reading.
 - Streaming Networks with Sequential Reading = StrSort.
- Streaming Networks vs. External Memory.
 - Issue is random access to external memory.
 - External memory with sequential access to each file = Streaming Networks.
 - External memory with sequential access to each file + only one File open for reading at any time = StrSort.
- The models are “robust” in that they describe the same computational classes. Differences come from real technology concerns.

Components

1. Infrastructure components should be generic and simple to use.
2. There should not be too many. Each should be reliable.
3. These components should talk to each other via the data store.

System Building.

- Software for data mining runs at different speeds and requires different resources.
 - Buffering and queues necessary for efficient execution.
 - Streaming Networks are therefore a natural model.
- One producer, many consumers.
 - Case in point, crawling. This has to be asynchronous.
- Streaming I/O.
 - Stream data through small chunks of code.
 - Compute nodes and data mining code is unreliable.
 - Do not want to lose work. System is responsible for state.
- Sorting is a very useful basic feature.
 - It should be supported by the system, and optimized to the hardware platform.